ANNALS
OF
DISCRETE
MATHEMATICS
1

studies in integer programming

Edited by
P.L. Hammer, E.L. Johnson,
B.H. Korte and G.L. Nemhauser

NORTH-HOLLAND

STUDIES IN
INTEGER PROGRAMMING

annals of discrete mathematics

Managing Editor

Peter L. HAMMER, University of Waterloo, Ont., Canada

Based on materials presented at the Workshop on Integer Programming, Bonn, 8-12 September 1975, organised by the Institute of Operations Research (Sonderforschungsbereich 21), University of Bonn. Sponsored by IBM Germany.

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

ANNALS OF DISCRETE MATHEMATICS 1

STUDIES IN INTEGER PROGRAMMING

Edited by

P.L. HAMMER, University of Waterloo, Ont., Canada
E.L. JOHNSON, IBM Research, Yorktown Heights, NY, U.S.A.
B.H. KORTE, University of Bonn, Federal Republic of Germany
G.L. NEMHAUSER, Cornell University, Ithaca, NY, U.S.A.

1977

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

Reprinted from the journal *Annals of Discrete Mathematics*, Volume 1

North-Holland ISBN for this Volume: 0 7204 [illegible]

Published by

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

Sole distributors for the U.S.A. and Canada:

ELSEVIER NORTH-HOLLAND, INC.
52 VANDERBILT AVENUE
NEW YORK, NY 10017

Printed in The Netherlands

PREFACE

This volume constitutes the proceedings of the Workshop on Integer Programming that was held in Bonn, September 8–12, 1975. The Workshop was organized by the Institute of Operations Research (Sonderforschungsbereich 21), University of Bonn and was generously sponsored by IBM Germany. In all, 71 participants from 14 different countries took part in the Workshop.

Integer programming is one of the most fascinating and difficult areas of mathematical optimization. There are a great many real-world problems of large dimension that urgently need to be solved, but there is a large gap between the practical requirements and the theoretical development. Since combinatorial problems in general are among the most difficult in mathematics, a great deal of theoretical research is necessary before substantial advances in the practical solution of problems can be expected. Nevertheless the rapid progress of research in this field has produced mathematical results significant in their own right and has also borne substantial fruit for practical applications. We believe that this will be adequately demonstrated by the papers in this volume.

The 37 papers appearing in this volume cover a wide spectrum of topics in integer programming. The volume includes works on the theoretical foundations of integer programming, on algorithmic aspects of discrete optimization, on specific types of integer programming problems, as well as on some related questions on polytopes and on graphs and networks.

All the papers have been carefully refereed. We express our sincere thanks to all authors for their cooperation, to the referees for their useful support, to numerous participants for stimulating discussions, and to the editors of the Annals of Discrete Mathematics for their willingness to include this volume in their new series.

Bonn, 1976 *The Program Committee*

F. Schweitzer
IBM Germany

P.L. Hammer
E.L. Johnson
B.H. Korte
G.L. Nemhauser

CONTENTS

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 1–11

REDUCTION AND DECOMPOSITION OF INTEGER PROGRAMS OVER CONES

Achim BACHEM

Institut für Ökonometrie und Operations Research, Universität Bonn, Nassestraße 2, D-53 Bonn, F.R.G.

Received: 1 August 1975
Revised: 7 November 1975

We consider the problem

$$(\dagger)\qquad \begin{aligned} &\min c'x\\ &\text{s.t. } Nx + By = b\\ &\quad x \in \mathbf{N}^r,\ y \in \mathbf{Z}^n \end{aligned}$$

where N is an (m, r), B an (m, n) integer matrix, and $b \in \mathbf{Z}^m$. In Section 2 we characterize all solutions $x \in \mathbf{Z}^r$ of (†) by an explicit formula and give as a corollary a minimal group representation of equality restricted integer programs, where some of the nonnegativity restrictions are relaxed. In Section 3 we discuss decomposing integer programs over cones in case the matrix N has special structure.

1. Introduction

We consider the problem

$$\begin{aligned} &\min c'x\\ &\text{s.t. } Nx + By = b\\ &\quad x \in \mathbf{N}^r,\ y \in \mathbf{Z}^n \end{aligned} \tag{1.1}$$

where N is an (m, r) and B an (m, n) integer matrix. As B is an arbitrary (m, n) integer matrix, the convex hull of the feasible set of (1.1) is a generalized corner polyhedron, that is an equality restricted integer program, where the nonnegativity restriction of some of the variables are relaxed. To give a group representation of the problem, we reformulate (1.1) as a congruence problem.

$$\begin{aligned} &\min c'x\\ &\text{s.t. } Nx \equiv b \mod B\\ &\quad x \in \mathbf{N}^r \end{aligned} \tag{1.2}$$

where we define $Nx \equiv b \pmod B$, iff there is a $\lambda \in \mathbf{Z}^n$, such that $Nx = b + B\lambda$ holds. To set this definition in a more general framework we have to introduce the concepts of Smith and Hermite normal form.

Definition. *If B is an (m, n) integer matrix, we denote by $S(B)$ and $H(B)$ the Smith and Hermite normal form of B, $S^*(B)$ and $H^*(B)$ denotes the nonsingular part of $S(B)$, $H(B)$ resp. The unimodular matrices which transform B into Smith normal form are denoted by U_B, K_B and the projection matrices, which eliminate the nonsingular part $S^*(B)$ of $S(B)$ are denoted by W_B, V_B. Thus we have $S^*(B) = W_B\, U_B\, B\, K_B\, V_B$.*

Sometimes it is advantageous to look at congruences from an algebraic point of view, that is to look at the definition of $a := x (\equiv \bmod \alpha)$[1] as an image of the function $a := h_\alpha(x) = x - \alpha\lfloor x/\alpha \rfloor$ (where "$\lfloor x \rfloor$" denotes the integer part of x). For (m, n) matrices B with rank $(B) \in \{m, n\}$ the scalar α is replaced in the above formula and we get the generalized form as

$$h_B(x) := x - \bar{B}\lfloor \bar{B}^+ x \rfloor$$

where $\bar{B}$ denotes the Hermite form $H(B)V_B$ of B (the zero colums of $H(B)$ are omitted) and where $\bar{B}^+$ denotes the Moore–Penrose inverse of $\bar{B}$. In fact we have

Proposition (1.3). *Let G be an additive subgroup of $\mathbf{Z}^m$. The map $h_B : G \to h_B(G)$ is a homomorphism onto $(h_B(G), \oplus)$ with kernel $(h_B) = \{x \in G \mid x = B\lambda, \lambda \in \mathbf{Z}^n\}$, and $x \oplus y := h_B(x + y)$.*

Remark (1.4). Obviously

$$a \equiv x (\equiv \bmod B)$$

$$\Longleftrightarrow a = x - B\lambda \quad \text{for some } \lambda \in \mathbf{Z}^n$$

$$\Longleftrightarrow a - x \in \text{kernel}(h_B) \quad \text{holds}$$

and so problem (1.2) is equivalent to

$$\min c'x$$

$$\bigoplus_i h_B(N_i) \cdot x_i = h_B(b), \tag{1.5}$$

$$x_i \in \mathbf{N},$$

where N_i denotes the ith column of the matrix N and "$=$" is the group equation in the group $G(B) := h_B(\mathbf{Z}^m)$.

Proof of Proposition (1.3). Since $\bar{B}$ has maximal column range, $\bar{B}'\bar{B}$ is regular, and we have

[1] "$:=$" means that the left side of the equation will be defined.

$$\bar B^+ \bar B = (\bar B' \bar B)^{-1} \bar B' \bar B = I.$$

So we conclude

$$\begin{aligned} h_B(x) + h_B(y) &= h_B(x) + h_B(y) - \bar B[\bar B^+(x+y) - ([\bar B^+ x] + [\bar B^+ y])] \\ &= x + y - \bar B[\bar B^+(x+y)] \\ &= h_B(x+y), \end{aligned}$$

hence h_B is a homomorphism. Let $x \in \mathrm{kernel}(h_B)$, that means $x = \bar B[\bar B^+ x]$. If we denote $b := [\bar B^+ x] \in \mathbf{Z}^r$ and $a := (b', 0, \ldots)'$ we conclude $x = H(B)a$ and $x = Bc$ where $c = Ka$, here K denotes the unimodular right multiplicator of $H(B)$. Let now $x = Ba$ with $a \in \mathbf{Z}^s$, that means $x = \bar B b$, $b \in \mathbf{Z}^r$. With $\bar B^+ x = b$ we conclude $h_B(x) = x - \bar B[\bar B^+ x] = \bar B b - \bar B b = 0$ which completes the proof.

Clearly problem (1.5) is a group problem over the group $G(B)$, which is not necessarily of finite order (it depends obviously on the rank of B). If we follow the usual definition of equivalent matrices (cf. [5]), that is the (m, n) integer matrix A and the (r, s) integer matrix B are equivalent iff they have the same invariant factors (apart from units), we get a slight generalization of a well known fact:

Remark (1.6). The groups $G(A)$ and $G(B)$ are isomorphic iff the matrices A and B are equivalent and $m - \mathrm{rank}(A) = r - \mathrm{rank}(B)$ holds.

Using this result it is easy to give a formula for the number of different (nonisomorphic) groups $G(B)$, where the product of invariant factors of the (m, n) matrices B is fixed. This number is well known for regular (m, m) integer matrices B. Here we are going to treat the general case.

Definition. Let B be an (m, n) integer matrix. We call the product of the invariant factors of B the *invariant* of B ($\mathrm{inv}(B)$) which coincides with the determinant of B in case B is a square nonsingular matrix.

If $d = \prod_{j=1}^k p_j^{r_j}$ is a representation of $d = \mathrm{inv}(B)$ as a product of prime factors and p a function from $\mathbf{N}^2$ into $\mathbf{N}$ defined recursively as

$$p(n, m) := \begin{cases} p(n, n), & 1 \le n \le m, \\ p(n, m-1) + p(n-m, m), & n \ge m \ge 1, \end{cases}$$

$p(0, m) := 1$, $p(n, 0) := 0$ ($n, m \in \mathbf{N}$), we define

$$K(d) := \sup_{m \in \mathbf{N}} \prod_{j=1}^k p(r_j, m)$$

$$L(d, m) := \sum_{l=1}^m \prod_{j=1}^k p(r_j, l).$$

Proposition (1.7). *The number of nonisomorphic groups $G(B)$, where B varies over all (m, n) integer matrices ($m, n \in \mathbf{N}$) with maximal row rank and invariant d, equals the integer number $K(d)$.*

The number of nonisomorphic groups $G(B)$, where B varies over all (m,n) integer matrices $(n \in \mathbb{N})$ with $\operatorname{rank}(B) \in \{m, n\}$ and invariant d, equals $L(d,m)$.

Notice that $K(d)$ is a finite number, though we consider all (m,n) integer matrices B with $m, n \in \mathbb{N}$. If we compute the numbers $K(d)$ and $L(d,m)$ for d's between 1 and 10[illegible], we note that $0 \leq K(d) \leq 10$ in 95% of the cases, that is the group $G(B)$ is more or less determined by $d = \operatorname{inv}(B)$.

Proof of Proposition (1.7). Two groups are isomorphic iff the generating matrices are equivalent and the rank condition holds (cf. Remark (1.6)). Proving the first part of the proposition we have only to deal with maximal row rank matrices and using Remark (1.4) we can restrict ourselves to square matrices, because $h_B(x)$ is defined in terms of $H^*(B)$ and this an (m,n) integer matrix with $\det H^*(B) = \operatorname{inv}(B)$. Because of the divisibility property of the invariant factors of an (m,m) integer matrix it suffices now to compute the number of different representations of the exponents of a prime factor presentation of the determinant $d = \det B$ as a sum of m nonnegative integers. In fact this number equals $p(e_i, m)$ (cf. (2)) and moreover $\mathrm{H}(d)$ is finite because

$$e_0 := \max_{i=1}^{k} e_i$$

leads to

$$\prod_{i=1}^{k} p(e_i, e_0 + k) = \prod_{i=1}^{k} p(e_i, e_0) \quad (k \in \mathbb{N}).$$

To prove the second part of the proposition we first note that $\operatorname{rank}(B) \leq m$. Since two groups $G(A)$ and $G(B)$ with matrices having both less than m columns, cannot be isomorphic, the second statement follows obviously from the first one.

2. Minimal group representation

We have seen that (1.5) is a group problem, namely of the group $G(B)$. In fact this is the group which will usually be considered in the asymptotic integer programming approach (cf. (3)), whereas the actual underlying group of (1.5) is the group

$$G(N/B) := \{h_B(x) / x = N\lambda, \lambda \in \mathbb{Z}^n\}$$

which is a subgroup of $G(B)$ generated by the columns of the matrix N. From a computational point of view the group $G(N/B)$ is more difficult to handle than the group $G(B)$ (though it has less elements), because there is no proper respresentation of $G(N/B)$. From this reason here we are going to find a $\bar{B} \in \mathbb{N}^{m}$ which will be defined in terms of N and B, such that the group $G(N/B)$ is isomorphic to

G (diag(δ)). Clearly this is a minimal group representation of problem (1.5) and as a corollary we get the order of $G(N/H)$ by

$$\prod_{i=1}^{m} \delta_i .$$

First we want to give some results concerning congruences which will be used later, they seem to be of general interest, though.

Theorem (2.1). *Let B be an (m,n) integer matrix with rank $(B)=m$, N an (m,s) integer matrix, $b \in \mathbb{Z}^s$ and $A := (N, B)$. The system of congruences*

$$Nx \equiv Nb \mod B$$

$$x \text{ integer}$$

has a solution iff $S^*(A)^{-1} V_A U_A b$ is integer. In this case, all solutions are of the form

$$x \equiv b \mod H$$

$$x \text{ integer}$$

where $H := (K_N V_M W_M L, R)$. Here we denote by $L := S^*(A)^{-1} U_A N$, $M := S^*(A)^{-1} U_A B$ and R denotes the last $s-k$ columns of K_N, where $k := \operatorname{rank}(N)$.

Proof. Without loss of generality we set $b=0$. It is easy to see that $S^*(M, L)$ equals an (m,m) identity matrix I^m, so we conclude

$$S(S(M), U_M L) = (I^m, 0_{m,s})$$

With $\operatorname{diag}(t_1, \dots, t_k) := S^*(M)$, $t_{k+i} := 0$ $(i = 1, \dots, m-k)$ and $D := U_M L$ we get immediately

(†) $\qquad \gcd(t_i, d_i) = 1 \quad i = 1, \dots, m,$

where $d_i := \gcd(D_{ij}, j = 1, \dots, n)$ $(i = 1, \dots, m)$.

Obviously the system

$$Nx \equiv 0 \mod B$$

$$x \text{ integer}$$

is equivalent to the system

$$\begin{pmatrix} S^*(M) & 0_{m,n-m} \\ 0_{n-m,m} & \end{pmatrix} y \equiv 0 \mod U_M L,$$

$$y \text{ integer},$$

and using (†) it is also equivalent to

$$(S^*(M), 0_{m,n-m})\, y \equiv 0 \mod W_M U_M L,$$

$$y \text{ integer}.$$

Let $y = (y_1', y_2')'$ be a $(k, s-k)$ partition of y, then we get

$$S^*(M)y_1 = 0 \mod W_0 U_0 L,$$

$$y_1, y_2 \text{ integer}.$$

Let K_i $(i = 1, \dots, k)$ be unimodular matrices, which transform the ith row of $\hat{D} := W_M U_M L$ into $(d_i, 0, \dots, 0)$. Using

$$E_i := K_i \operatorname{diag}(1, \dots, 1, t_i^{-1}, 1, \dots, 1) K_i^{-1}$$

$i = 1, \dots, m$ we define

$$F := \prod_{i=1}^{m} E_i$$

By induction on i one can easily show that

$$\operatorname{diag}(1, \dots, 1, t_{i+1}, \dots, t_k) y_1 = \hat{D} \prod_{j=1}^{i} E_j z$$

$$y_2, \prod_{j=1}^{i} E_j z \text{ integer}$$

is equivalent (for all $i = 1, \dots, m$) to

(*) $$S^*(M)y_1 = 0 \mod H$$

$$y_1, y_2 \text{ integer}$$

so that

$$y_1 = \hat{D} F z$$

$$y_2, Fz \text{ integer}$$

is equivalent to (*).

Since F^{-1} is an integer matrix and $x = K_M y$, the equation

$$x = (K_M V_M y_1 + R y_2)$$

completes the proof.

Theorem (3.2). *With the notations of theorem* (2.1) *we get*

(i) $S^*(L) = S(A)^{-1} U_A U_B^{-1} S^*(B)$

(ii) $S^*(H) = T^{-k} + \operatorname{diag}(t_{k+1}, \dots, t_m)$

where $S^*(L) =: \operatorname{diag}(t_1, \dots, t_m)$.

Proof. Because of

$$L = S^*(A)^{-1} U_A U_B^{-1} U_B B,$$

(i) follows immediately from the equation

$$S^*(L) = S^*(LK_B) = S^*(LK_B V_B).$$

Let

$$P := \begin{pmatrix} 0_{k,s-k} \\ I^{s-k} \end{pmatrix}$$

where I^{s-k} denotes an $((s-k),(s-k))$ identity matrix. Because of $H = K_{nr}(W_{kk}U_{kk}L, P)$, we conclude $S^*(H) = S^*(W_{kk}U_{kk}L, P)$, that is

$$S^*(H) = \begin{pmatrix} I^{s-k} & 0, \ldots \\ 0, \ldots & S^*(QL) \end{pmatrix}$$

where Q denotes the first k rows of U_{kk}.

From the proof of theorem (2.1) we know that

$$S^*(L) = S^*(H(U_{kk}L)) = \operatorname{diag}(t_1, \ldots, t_k),$$

so

$$S^*(QL) = \operatorname{diag}(t_{s-k+1}, \ldots, t_k)$$

which completes the proof.

Now we are able to give an isomorphic representation of the subgroup $G(N/B)$.

Theorem (2.3). *Let B be an (m,n) and N an (m,r) integer matrix with $\operatorname{rank}(B) = m$. Then we get*

$$G(N/B) \cong G(S^*(E)),$$

that means the group $G(N/B)$ is isomorphic to the group $G(S^*(E))$, where $E := W_{kk}U_{kk}L$ and $L := S^*(N,B)^{-1}U_{(N,B)}N$, $M := S^*(N,B)^{-1}U_{(N,B)}B$.

Corollary (2.4).

$$\Theta := U_E\, S^*(M)^{-1} W_{kk}U_{kk}\, S^*(N,B)^{-1} U_{(N,B)}$$

is an isomorphism from $G(N/B)$ to $G(S^*(E))$.

Corollary (2.5). *The order of $G(N/B)$ equals*

$$\frac{\operatorname{inv}(B)}{\det(S^*(N,B))}.$$

Proof of Theorem (2.3). Let K be a unimodular matrix, so that NK is up to permutations of rows in Hermite normal form. Let $\bar{N}$ be the matrix NK without the zero columns. Obviously we have $G(N/B) = G(\bar{N}/B)$. Let

$$[\bar{N}] := \{x \in \mathbf{Z}^m \mid x = \bar{N}y \quad \text{for any } y \in \mathbf{Z}^r\}$$

be a subgroup of $(\mathbf{Z}^m, +)$. Because $h_B : [\bar{N}] \to h_B([\bar{N}])$ is a homomorphism (Proposition 1.5) $G(\bar{N}/B)$ is isomorphic to the factor group

$$\{\bar{N}\}/\mathrm{kernel}(h_B)$$

where $\mathrm{kernel}(h_B)=\{x\in\{\bar{N}\} \mid x\equiv 0 \bmod B\}$.

With Theorem (2.1) we conclude

$$\mathrm{kernel}(h_B)=\{x\in \mathbb{Z}^n \mid x=\bar{N}y,\ y\equiv 0 \bmod K_N W_N U_N L \text{ for all } y\in \mathbb{Z}^k\}.$$

Let

$$f := S^*(M)^{-1} W_N U_N B L^{-1}.$$

Then

$$f:\{\bar{N}\}\to \mathbb{Z}^k$$

is an isomorphism and $f(\mathrm{kernel}(h_B))=\{z\in \mathbb{Z}^k \mid z\equiv 0 \bmod W_N U_N L\}$. Thus we get

$$\{N\}/\mathrm{kernel}(h_B)\cong \mathbb{Z}^k/\mathrm{kernel}(h_L)$$

and because U_N is also an isomorphism we get the isomorphism

$$G(N/B)\cong G(S^*(L))$$

The corollaries follow immediately from Theorem (2.3) in conjunction with Theorem (2.2).

3. Partitioning of integer programs over cones

The computational effort to solve the problem

$$\begin{aligned} &\min c'x \\ \text{s.t. } & Nx+By=b \\ & x\in \mathbb{N},\ y\in \mathbb{Z}^k \end{aligned} \tag{3.1}$$

usually grows rapidly according to the determinant of B. It is therefore sometimes advantageous to decompose the problem into smaller subproblems and to link the optima of the subproblems to a solution of the masterproblem. We give now two examples of decomposing problem (3.1) in case the matrix N is of the form

$$N=\begin{bmatrix} N_1 & & & 0 \\ & \cdot & & \\ & & \cdot & \\ 0 & & & N_r \end{bmatrix} \tag{3.2}$$

or

$$N=\begin{bmatrix} A_1 & \cdots & \cdots & A_r \\ N_1 & & & \\ & \cdot & & \\ & & \cdot & \\ 0 & & & N_r \end{bmatrix} \qquad b=\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_r \end{bmatrix} \tag{3.3}$$

To simplify notation let $B = S^*(B)$, i.e. B is given as a diagonal matrix. (Otherwise we have to impose some special structure on U_B.)

Let us denote the set of feasible solutions of problem (3.1) by

$$SG(N, b/B) := \{x \in \mathbf{N}^r \mid Nx - b \in \operatorname{kernel}(h_B)\}.$$

Let N be an (m, r) integer matrix of form (3.2), let $b_i(x) := h_B(b - N_1 x)_{I_i}$ where I_i corresponds to the row indices of the submatrix N_i and let us denote by

$$z_i(b_i(y)) := \begin{cases} \infty & \text{if } b_i(y) \notin G(N_i/B_{ii}), \\ \min c_i' x_i & \\ x_i \in SG(N_i, b_i(y)/B_{ii}) & \text{otherwise}, \end{cases}$$

the optimal value of the subproblems.

Proposition (3.4). *The programs*

$$\min c'x$$
$$x \in SG(N, b/B), \tag{3.5}$$

$$\min c_1 y + \sum_{i=2}^{r} z_i(b_i(y))$$
$$y \in \mathbf{N} \tag{3.6}$$

are equivalent.

Proof. Let $x_i(y)$ be the minimand corresponding to the optimal value $z_i(b_i(y))$. Let y be optimal in (3.6) and assume that there is an $\hat{x} \in SG(N, b/B)$, $(\hat{x} \neq x := (y, x_2(y), \ldots, x_r(y)))$ such that $c'\hat{x} < c'x$.

Let $\hat{x} = (\hat{y}, \hat{x}_2, \ldots, \hat{x}_r)$, where $\hat{y}$ are the components corresponding to N_1. Because $\hat{x}_i$ are feasible, we get

$$c_i'\hat{x}_i \geq \min c_i x_i = c_i'\bar{x}_i \qquad i = 2, \ldots, r$$
$$x_i \in SG(N_i, b_i(\hat{y})/B_{ii})$$

and the contradiction

$$c'\hat{x} = c_1\hat{y} + \sum_{i=2}^{r} c_i'\hat{x}_i \geq c'x = \min\left\{c_1'y + \sum_{i=2}^{r} z_i(b_i(y)) \,\middle|\, y \in \mathbf{N}\right\}$$

proves one part of the proposition, however the reverse direction is trivial.

Let again N be an (m, r) integer matrix which has form (3.3) and define $z_1(x_2, \ldots, x_r) := \min c_1 x_1$
s.t.

$$x_1 \in SG\left[\begin{pmatrix} A_1 \\ N_1 \end{pmatrix}, \begin{pmatrix} b_0 - \sum_{i=2}^{r} A_i x_i \\ b_1 \end{pmatrix} \Big/ B_1\right],$$

$$z_i(x_i, \ldots, x_r) := \min c_i x_i + z_{i-1}(x_i, \ldots, x_r)$$
$$x_i \in SG(N_i, b_i/B_{ii}), \quad i = 2, \ldots, r$$

as the optimal value of the subproblems.

Proposition (3.5). *The programs*

$$\min c'x \tag{3.6}$$
$$x \in SG(N, b/B)$$

and

$$\min c_r x_r + z_{r-1}(x_r) \tag{3.7}$$
$$x_r \in SG(N_r, b_r/B_r)$$

are equivalent.

Proof. If we denote by

$$c'\bar{x} := \min c'x$$
$$x \in SG(N, b/B)$$

we obviously get

$$c_1\bar{x}_1 = \min c_1 x_1$$
$$x_1 \in SG\left[\begin{pmatrix} A_1 \\ N_1 \end{pmatrix}, \begin{pmatrix} b_1 - \sum_{i=2}^{r} A_i\bar{x}_i \\ b_1 \end{pmatrix} \Big/ B_1\right]$$

which yields in the same way

$$\min c_i x_i + z_{i-1}(x_i, \bar{x}_{i+1}, \ldots, \bar{x}_r) = \sum_{j=1}^{i} c_j\bar{x}_j$$

for all $i > 1$, because

$$c_k\bar{x}_k + z_{k-1}(\bar{x}_k, x_{k+1}, \ldots, x_r) < \sum_{j=1}^{r} c_j x_j$$

implies

$$c'\bar{x} > \sum_{i=1}^{r} c_i\bar{x}_i - c_k\bar{x}_k.$$

So we get the result

$$c'\bar{x} = \min c_r x_r - z_{r-1}(x_r)$$
$$x_r \in SG(N_r, b_r/B_r),$$

which completes the proof.

The computational experience with algorithms canonically based on Propositions (3.4) and (3.5) is up to now limited to some of the Bradley–Wahi [1] test examples, which have determinants greater than 1,000,000. The results are very promising in the sense that it is possible to solve "cone problems" of such large order. The complete computational results together with comparisons of existing group algorithms will be the subject of a following paper.

Acknowledgment. I wish to acknowledge the interesting discussions I had with E.L. Johnson on the subject of this paper. The paper has been revised substantially while he was a visiting professor at the University of Bonn.

References

[1] G.H. Bradley and P.N. Wahi, Integer Programming Test Problems, Report No. 28, Yale University, New Haven, December 1969.

[2] L. Comtet, Advanced Combinatorics (Reidel, Dordrecht, 1974).

[3] R.E. Gomory, On the Relation between Integer and Non-Integer Solutions to Linear Programs, Proc. Nat. Acad. Sci. 53 (1965) 260–265.

[4] M. Marcus and E.E. Underwood, A Note on the Multiplicative Property of the Smith Normal Form, J. of Res. of the Nat. Bureau of Standards-B, 76B (1972) 205–206.

[5] M. Newman, Integral Matrices (Academic Press, New York, 1972).

[6] M. Newman, The Smith Normal Form of a Partitioned Matrix, J. of Res. of the Nat. Bureau of Standards-B, Vol. 78B (1974) 3–6.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 13-47

SOME VALID INEQUALITIES FOR THE SET PARTITIONING PROBLEM*

Egon BALAS
Carnegie-Mellon University

We introduce a family of inequalities derived from the logical implications of set partitioning constraints and investigate their properties and potential uses. We start with a class of homogeneous canonical inequalities that we call elementary, and discuss conditions under which they are (a) valid, (b) cutting planes, (c) maximal, and (d) facets or improper faces of the set partitioning polytope. We give two procedures for strengthening nonmaximal valid elementary inequalities. Next we derive two nonhomogeneous equivalents of the elementary inequalities, which are of the set packing and set covering types respectively. Using the first of these equivalents, we introduce a "strong" intersection graph, a supergraph of the (common) intersection graph, whose facet generating subgraphs (cliques, odd holes, etc.) give rise to valid inequalities for the set partitioning problem. These inequalities subsume or dominate the similar inequalities that one can derive for the associated set packing problem. One subclass can be used to enhance orthogonality tests in implicit enumeration or column generating algorithms. Further, we introduce two types of composite inequalities, obtainable by combining elementary inequalities according to specific rules, and some related inequalities obtainable directly from the set partitioning constraints. These inequalities provide convenient primal all integer cutting planes that offer a greater flexibility and are usually stronger than the earlier cuts which do not use the special structure of the set partitioning problem. In the final section we discuss a primal algorithm which uses these cuts in conjunction with implicit enumeration.

1. Introduction

Set partitioning is one of those combinatorial optimization problems which have wide-ranging practical applications and for which no polynomially bounded algorithm is available. Though both implicit enumeration and cutting plane algorithms have been reasonably successful on this problem, the practical importance of solving larger set partitioning models than we can currently handle makes this a very lively research area (see [6] for a recent survey of theoretical results and algorithms, and a bibliography of applications).

In this paper we introduce a family of valid inequalities derived from the logical implications of the set partitioning constraints, and investigate their properties and potential uses. We first define some basic concepts, then at the end of this section we outline the content of the paper.

The set partitioning problem can be stated as

* This research was supported by the National Science Foundation under Grant #GP 37510X1 and by the U.S. Office of Naval Research under contract N00014-67-A-0314-0007NR.

$$\min\{cx \mid Ax = e, x_j = 0 \text{ or } 1, j \in N\}$$

where $A = (a_{ij})$ is an $m \times n$ matrix of 0's and 1's, e is an m-vector of 1's, $N = \{1, \ldots, n\}$. We will denote by a_j the jth column of A, and assume that A has no zero row and no zero column. Also, we will write $M = \{1, \ldots, m\}$.

The convex hull and the dimensions of a set S, and the vertex set of a polytope T, will be denoted by conv S, dim S and vert T respectively.

Denoting by "conv" the convex hull, we will call

$$\mathrm{P} = \operatorname{conv}\{x \in \mathbf{R}^n \mid Ax = e, x_j = 0 \text{ or } 1, j \in N\}$$

the *set partitioning polytope*, and denote the linear programming relaxation of P by

$$\mathrm{LP} = \{x \in \mathbf{R}^n \mid Ax = e, x \geq 0\}.$$

Clearly, vert $\mathrm{P} = \mathrm{P} \cap \{0,1\}^n$.

We will also refer to

$$\bar{\mathrm{P}} = \operatorname{conv}\{x \in \mathbf{R}^n \mid Ax \leq e, x_j = 0 \text{ or } 1, j \in N\},$$

the *set packing polytope* associated with P.

Whenever $\mathrm{P} \neq \emptyset$, we have

$$\dim \mathrm{P} \leq \dim \mathrm{LP} = n - r(A)$$

where $r(A)$ is the rank of A.

An inequality

$$\pi x \geq \pi_0 \tag{1}$$

satisfied by all $x \in \mathrm{P}$ is called *valid* for P. A valid inequality (1) such that

$$\pi x = \pi_0 \tag{1'}$$

for exactly $k+1$ affinely independent points $x \in \mathrm{P}$, $0 \leq k \leq \dim \mathrm{P}$, defines a k-dimensional *face* of P and will itself be called a face (though since $\dim \mathrm{P} < n$, a given face can be defined by more than one inequality). If $k < \dim \mathrm{P}$, the face is *proper*, otherwise it is *improper*. In the latter case, the hyperplane defined by (1') contains all of P, and is called *singular*.

A valid inequality (1) is a *cut* or *cutting plane*, if it is violated by some $x \in \mathrm{LP} \setminus \mathrm{P}$. A face of P, whether proper or not, may or may not be a cutting plane. If $\dim \mathrm{P} = \dim \mathrm{LP}$, then the affine hull of P is the same as that of LP; hence any hyperplane which contains all of P, also contains all of LP, and therefore no improper face of P is a cutting plane. If $\dim \mathrm{P} < \dim \mathrm{LP}$, then improper faces of P may also be cutting planes.

Proper faces of maximal dimension are called *facets*. Evidently, P has faces (hence facets) if and only if $\dim \mathrm{P} \geq 1$, which implies $n \geq r(A)$. If $\dim \mathrm{P} = \dim \mathrm{LP}$, then the facets of P are of dimension $n - r(A) - 1$, i.e., each facet contains exactly $n - r(A)$ affinely independent points of P. Since $0 \notin$ P, these affinely independent points are linearly independent vectors.

A valid inequality (1) is *maximal* if for any $k \in N$ and any $\pi'_k > \pi_k$ there exists $x \in$ P such that

$$\pi'_k x_k + \sum_{j \in N \setminus \{k\}} \pi_j x_j > \pi_0.$$

This notion is the same as that of a minimal inequality (see Gomory and Johnson [12]; and, more recently Jeroslow [13]), except that here we find it more convenient to consider inequalities of the form $\leq$ rather than $\geq$, in order to have a nonnegative righthand side.

The following is an outline of the content of this paper.

We start (Section 2) with a class of homogeneous canonical inequalities that we call elementary, since all the subsequent inequalities can be built up from these first ones by various composition rules. The elementary inequalities, together with the 0–1 condition and the constraints $Ax \leq e$, imply the constraints $Ax \geq e$; but they also cut off fractional points satisfying $Ax = e$, $x \geq 0$. We discuss the conditions under which a given elementary inequality is (a) a cutting plane, (b) maximal, (c) a facet or an improper face of P.

When a given elementary inequality is not maximal, it can be strengthened. In Section 3 we discuss two systematic strengthening procedures for these inequalities.

In Section 4 we show that each elementary inequality is equivalent on LP to a set packing inequality and to each of several set covering inequalities. The first one of these equivalences suggests a graph-theoretical interpretation. We introduce a "strong" intersection graph of the matrix A defining P, and show that a set packing inequality is valid for P if and only if it corresponds to a complete subgraph of the strong intersection graph of A; and it is maximal if and only if this complete subgraph is a clique.

The next two sections deal with composite inequalities, obtained by certain rules from the elementary inequalities. These composite inequalities have the following property. Given an integer basic solution to the system $Ax = e$, $x \geq 0$, and a set S of nonbasic variables, none of which can be pivoted into the basis with a value of 1 without making the solution infeasible, there exists a composite inequality which can be used as a primal all-integer cut to pivot into the basis any of the variables in S without losing feasibility.

Finally, in Section 7 we introduce a class of inequalities which are satisfied by every feasible integer solution better than a given one, and which can be strengthened to a desired degree by performing implicit enumeration on certain subproblems. We then discuss a hybrid primal cutting plane/implicit enumeration algorithm based on these results.

Throughout the paper, the statements are illustrated on numerical examples.

2. Elementary inequalities

We shall denote

$$M_k = \{i \in M \mid a_{ik} = 1\}, \qquad \bar{M}_k = M \setminus M_k, \quad k \in N,$$

$$N_i = \{k \in N \mid a_{ik} = 1\}, \qquad \bar{N}_i = N \setminus N_i, \quad i \in M,$$

$$N_{ik} = \{j \in N_i \mid a_k a_j = 0\}, \quad i \in \bar{M}_k, \quad k \in N.$$

N_{ik} is the index set of those columns a_j orthogonal to a_k and such that $a_{ij} = 1$. Since $a_{ik} = 0$ (as a result of $i \in \bar{M}_k$), $x_k = 1$ implies that at least one of the variables x_j, $j \in N_{ik}$, must be one.

Valid inequalities of the form

$$x_k - \sum_{j \in Q} x_j \leq 0,$$

where $Q \subseteq N_{ik}$ for some $i \in \bar{M}_k$, will be called *elementary*. They play a central role as building blocks for all the inequalities discussed in this paper. These elementary inequalities are *canonical* in the sense of [4] (i.e., they have coefficients equal to 0, 1 or -1), hence each of them is parallel to a $(n - |Q| - 1)$-dimensional face of the unit cube.

Remark 2.1. *The slack of an elementary inequality is a 0–1 variable.*

Proof. Since $Q \subseteq N_{ik} \subseteq N_i$ for some $i \in M$, the sum of the variables indexed by Q cannot exceed 1.

Proposition 2.1. *For every $k \in N$ and $i \in \bar{M}_k$, the inequality*

$$x_k - \sum_{j \in N_{ik}} x_j \leq 0 \tag{2}$$

is satisfied by all $x \in P$.

Proof. From the definition of N_{ik}, for every $x \in \operatorname{vert} P$, $x_k = 1$ implies $x_j = 1$ for at least one $j \in N_{ik}$. But this is precisely the condition expressed by (2); thus (2) is satisfied by all $x \in \operatorname{vert} P$, hence by all $x \in P$. $\Box$

Remark 2.2. *The number of distinct inequalities (2) is at most $\sum_{k \in N} |\bar{M}_k|$.*

Proof. There is one inequality (2) for every zero entry of the matrix A, but some of these inequalities may be identical.

The converse of Proposition 2.1 is not true in general, i.e., a 0–1 point satisfying all inequalities (2) need not be in P, as one can easily see from the counterexample offered by $\bar{x}$ such that $\bar{x}_j = 1$, $\forall j \in N$. However, a weaker converse property holds

Proposition 2.2. *Any $x \in \{0,1\}^n$, $x \neq 0$, which satisfies $Ax \le e$ and all the inequalities (2), also satisfies $Ax = e$.*

Proof. Let $\bar{x} \in \{0,1\}^n$, $\bar{x} \neq 0$, be such that $A\bar{x} \le e$, $A\bar{x} \neq e$. Then there exists $i \in M$ such that $\bar{x}_j = 0$, $\forall j \in N_i$. Further, since $\bar{x} \neq 0$, there exists $k \in \bar{N}_i$ such that $\bar{x}_k = 1$. Therefore $\bar{x}$ violates the inequality

$$x_k - \sum_{j \in N_{ik}} x_j \le 0,$$

since $N_{ik} \subseteq N_i$. □

Corollary 2.2.1. *Every nonzero vertex of $\bar{P}$ not contained in P is cut off by some inequality (2); and every inequality (2) cuts off some $x \in \bar{P} \setminus P$.*

Proof. Every $x \in \bar{P} \setminus P$ violates $Ax \ge e$; hence if it is a nonzero vertex of $\bar{P}$, according to Proposition 2.2 it violates some inequality (2). On the other hand, every inequality (2) cuts off the point $\bar{x} \in \bar{P}$ defined by $\bar{x}_k = 1$, $\bar{x}_j = 0$, $\forall j \in N \setminus \{k\}$. □

Proposition 2.3. *For $k \in N$, $i \in \bar{M}_k$ and $Q \subset N_{ik}$, the inequality*

$$x_k - \sum_{j \in Q} x_j \le 0 \tag{3}$$

is valid if and only if $x \in \operatorname{vert} P$ and $x_k = 1$ implies $x_j = 0$, $\forall j \in N_{ik} \setminus Q$.

Proof. *Necessity*: if $x \in \operatorname{vert} P$ and $x_k = x_j = 1$ for some $j \in N_{ik} \setminus Q$, then from Remark 2.1, $x_j = 0$, $\forall j \in Q$ (since $Q \subseteq N_{ik}$), and x violates (3).

Sufficiency: if $x \in \operatorname{vert} P$ and $x_k = 1$ implies $x_j = 0$, $\forall j \in N_{ik} \setminus Q$, then (3) is valid because (2) is valid. □

Next we illustrate the elementary inequalities on a numerical example.

Example 2.1. Consider the numerical example of [5], i.e., the set partitioning polytope with coefficient matrix A (where the blanks are zeroes):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1					1	1	1	1				1	1	
2	1	1								1	1				1
3		1	1			1	1	1				1			
4		1	1	1				1	1		1				
5			1	1	1		1			1		1	1		

For $k = 1$, $\bar{M}_1 = \{3,4,5\}$; $N_{31} = \{3,12\}$, $N_{41} = \{3,4\}$, $N_{51} = \{3,4,5,12\}$, and the inequalities (2) are

$$x_1 - x_3 - x_4 \leq 0$$

$$x_1 - x_3 - x_{12} \leq 0$$

$$x_1 - x_3 - x_4 - x_5 - x_{12} \leq 0,$$

where the last inequality is dominated by each of the other two and hence is redundant. Further, $x_1 = 1$ implies $x_4 = x_{12} = 0$ for any feasible partition (this can be seen by inspection; systematic checking of such implications is discussed in Section 3) and therefore each of the sets N_{i_1}, N_{i_2} and N_{i_3} can be replaced by $Q = \{3\}$, and each of the above inequalities can be replaced by

$$x_1 - x_3 \leq 0.$$

In the next section we discuss procedures for strengthening elementary inequalities of the type (3) (which subsumes (2)) by systematically reducing the size of the sets Q subject to the condition of Proposition 2.3.

As mentioned in Section 1, a valid inequality may or may not be a cut, i.e., may or may not be violated by some $x \in \mathrm{LP} \setminus \mathrm{P}$.

Proposition 2.4. *The (valid) inequality (3) is a cut if and only if there exists no $\theta \in \mathbf{R}^m$ satisfying*

$$\theta a_j \geq \begin{cases} -1 & j = k, \\ 1 & j \in Q, \\ 0 & j \in N \setminus Q, \end{cases} \tag{4}$$

$$\theta e \leq 0. \tag{5}$$

Proof. According to a classical result (see, for instance, [20, Theorem 1.4.4]), (3) is a consequence of the system $Ax = e$, $x \geq 0$ if and only if there exists $\theta \in \mathbf{R}^m$ satisfying (4) and (5). If (3) is a consequence of $Ax = e$, $x \geq 0$, it is clearly not a cut. Conversely, if (3) is not a cut, then it is satisfied by all $x \in \mathrm{LP}$, hence a consequence of $Ax = e$, $x \geq 0$. □

Next we address the question of when a given elementary inequality is undominated, i.e., maximal. First, if for some $j \in N$, $x \in \mathrm{P}$ implies $x_j = 0$, then clearly the coefficient of x_j can be made arbitrarily large without invalidating the given inequality. Therefore, without loss of generality, we can exclude this degenerate case from our statement.

Proposition 2.5. *Assume that the inequality (3), where $Q \subseteq N_i$ for some $k \in N$, $i \in \bar{M}_k$, is valid; and for every $j \in N$ there exists $x \in \operatorname{vert} \mathrm{P}$ such that $x_j = 1$. Then (3) is maximal if and only if*

(i) *for every $j \in Q$ there exists $x \in \operatorname{vert} \mathrm{P}$ such that $x_j = x_k = 1$;*

(ii) *for every $i \in \bar{N}_i \setminus \{k\}$ there exists $x \in \operatorname{vert} \mathrm{P}$ such that $x_i = 1$ and $x_h = x_k$, $\forall h \in Q$.*

Proof. This is a specialization of the statement that a valid inequality $\pi x \le \pi_0$ for a 0–1 polytope $T \subseteq \mathbb{R}^n$ is maximal if and only if for every $j \in N$ there exists $x \in T$ such that $x_j = 1$ and $\pi x = \pi_0$. □

If a valid inequality is not maximal, then at least one of its coefficients can be increased without cutting off any $x \in P$. In the case of an arbitrary polytope, this is all we know, and it is not true in general that more than one coefficient can be increased without invalidating the inequality. In the case of elementary inequalities for P, however, one can say more.

Corollary 2.5.1. *Assume that for every $j \in N$ there exists $x \in \operatorname{vert} P$ such that $x_j = 1$. Let (3) be a valid, but not maximal inequality, with $Q \subseteq N_k$ for some $k \in N$, $i \in M_k$, and let S_1, S_2 be the sets of those $j \in N$ for which conditions* (i) *and* (ii), *respectively, are violated. Then all $x \in P$ satisfy the inequality*

$$x_k - \sum_{j \in Q \cap S_1} x_j \le 0 \tag{6}$$

and the inequalities

$$x_k + \sum_{j \in S_2 \cap T} x_j - \sum_{j \in Q} x_j \le 0, \tag{7}$$

for every $T \subseteq \bar{N}_i \setminus \{k\}$ such that $a_{ih} a_{ij} \neq 0$, $\forall h, j \in T$.

Proof. The validity of (6) follows from Proposition 2.3 and the definition of S_1.

To prove the validity of (7), let $x \in \operatorname{vert} P$ be such that $x_k = 1$. Then $x_j = 0$, $\forall j \in S_2 \cap \bar{N}_i \setminus \{k\}$ (hence $\forall j \in S_2 \cap T$), since otherwise from the definition of S_2, $x_k = x_h = 1$ for some $h \in Q$, which is impossible. Further, from (3), $x_j = 1$ for some $j \in Q$. Hence (7) holds for all $x \in P$ such that $x_k = 1$.

Now let $x_k = 0$. From the definition of T, $x_j = 1$ for at most one $j \in T$; and from the definition of S_2, $x_j = 1$ for some $j \in S_2 \cap T$ implies $x_k \le x_h$ for some $h \in Q$, i.e., $x_h = 1$ for at least one $h \in Q$. Hence (7) also holds for all $x \in P$ such that $x_k = 0$. □

Clearly, if for some $S' \subset N$ the nondegeneracy assumption of Proposition 2.5 (and Corollary 2.5.1) is violated for all $j \in S'$, then the coefficient of each x_j, $j \in S'$, can be made arbitrarily large, in addition to the changes in the coefficients of x_j, $j \in S = S_1 \cup S_2$, justified by the Corollary.

From the above Corollary, nonmaximal elementary inequalities can be strengthened, provided we know S. In the following sections we give several procedures for identifying subsets of S.

Next we turn to the question of when a maximal elementary inequality is a face of maximal dimension, i.e., a facet or an improper face of P. This question is of interest since P is the intersection of the halfspaces defining its facets and improper faces. The next proposition gives a sufficient condition for an elementary inequality to be a facet or an improper face of P.

Proposition 2.6. *Suppose (3) is a maximal (valid) inequality for P, with $Q \subseteq N_k$ for some $k \in N$, $i \in \bar{M}_k$. Let $N' = N \setminus Q \cup \{k\}$, and*

$$P_{N'} = P \cap \{x \in \mathbf{R}^n \mid x_j = 0, \forall j \in Q \cup \{k\}\}.$$

Then $\dim P \geq \dim P_{N'} + q$, *where* $q = |Q|$.

If $\dim P = \dim P_{N'} + q$, *then (3) is an improper face of P.*

If $\dim P = \dim P_{N'} + q + 1$, *then (3) is either a facet or an improper face of P.*

Proof. Let $d = \dim P$, $d' = \dim P_{N'}$. Since (3) is maximal, for every $j \in Q$ there exists $x^j \in \operatorname{vert} P$ such that $x_j^j = x_k^j = 1$. Also, since $Q \subseteq N_k$, $x_h^j = 0$, $\forall h \in Q \setminus \{j\}$ for each of these q points x^j. With each point x^j, $j = 1, \ldots, q$, we associate a row vector $y^j \in \mathbf{R}^n$, obtained by permuting the components of x^j so that x_k^j comes first, and the components indexed by Q come next.

Further, let $z^j \in \mathbf{R}^{N'}$, $j = 1, \ldots, d' + 1$, be a maximal set of affinely independent vertices of $P_{N'}$, and let $y^{q+j} \in \mathbf{R}^n$, $j = 1, \ldots, d' + 1$ be row vectors of the form $y^{q+j} = (0, z^j)$, where 0 has q components. Clearly, each y^{q+j} is, modulo the permutation of components, a vertex of P. Then the matrix Y whose rows are the vectors y^j, $j = 1, \ldots, q + d' + 1$, is of the form

$$Y = \left[\begin{array}{c:c} X_1 & X_2 \\ \hdashline 0 & Z \end{array}\right]$$

where X_1 is the $q \times (q + 1)$ matrix

$$X_1 = \begin{bmatrix} 1 & 1 & & & \\ 1 & & 1 & & \\ \vdots & & & \ddots & \\ 1 & & & & 1 \end{bmatrix}$$

(the blanks stand for zeroes), Z is the $(d' + 1) \times (n - q - 1)$ matrix whose rows are the vectors z^j, $j = 1, \ldots, d' + 1$, 0 is the $(d' + 1) \times (q + 1)$ zero matrix, and X_2 is a $q \times (n - q - 1)$ matrix of zeroes and ones.

Since X_1 and Z are of full row rank, so is Y; and since Y has $q + d' + 1$ rows, it follows that P contains at least $q + d' + 1$ affinely independent points; hence $d \geq d' + q$.

If $d = d' + q$, then the $d' + q + 1$ rows of Y define a maximum-cardinality set of affinely independent points of P; and since each of these points satisfies (3) with equality, the same is true of every other point of P. Hence in this case (3) is an improper face of P.

If $d = d' + q + 1$, then there exists a point $x^* \in P$ which, together with the $d' + q + 1$ points corresponding to the rows of Y, forms an affinely independent set. If x^* also satisfies (3) with equality, then (3) is an improper face of P; otherwise (3) is a facet of P. $\square$

Example 2.2. In example 2.1, the inequalities (2) for $k = 1$ and $i = 3, 4$, i.e., the inequalities

$$x_1 - x_5 - x_3 \le 0, \qquad x_1 - x_5 - x_4 \le 0$$

are cutting planes, since each of them cuts off the fractional point $\bar{x}$ defined by $\bar{x}_1 = \bar{x}_3 = \bar{x}_6 = \frac{1}{2}$, $\bar{x}_9 = 1$, $\bar{x}_j = 0$ otherwise, but they are not maximal, since the conditions of Proposition 2.5 are violated for $j = 9, 12$ in the case of the first inequality and $j = 4$ in the case of the second one. Therefore, $x_1 - x_5 \le 0$ and $x_1 - x_3 + x_2 - x_{12} \le 0$ are both valid (Corollary 2.5.1). The inequality $x_1 - x_4 \le 0$ is maximal, since the assumptions and conditions of Proposition 2.5 are satisfied. It is also a facet of P, since the dimensionality condition of Proposition 2.6 is satisfied and the point $\bar{x}$ defined by $\bar{x}_7 = \bar{x}_{11} = \bar{x}_{12} = 1$, $\bar{x}_j = 0$ otherwise, does not lie on $x_1 - x_4 = 0$.

On the other hand, if P' is the set partitioning polytope obtained from P by removing the last column of A, then $x_1 - x_4 \le 0$ is an improper face of P' since $x \in P'$ implies $x_1 - x_4 = 0$.

3. Strengthening procedures

An inequality $\pi' x \le \pi_0$ is called *stronger* than $\pi x \le \pi_0$, if $\pi'_j \ge \pi_j$ for all j, and $\pi'_j > \pi_j$ for at least one j.

In this section we discuss two procedures for replacing a valid elementary inequality which is not maximal, with a stronger valid elementary inequality. The first procedure uses information from the other elementary inequalities in which x_k has a positive coefficient; the second one uses information from the elementary inequalities in which x_j has a positive coefficient for some $j \in Q_i$.

Proposition 3.1. *For some $k \in N$, let the index sets $Q_i \subseteq N_{ik}$, $i \in \bar{M}_k$, be such that the inequalities*

$$x_k - \sum_{j \in Q_i} x_j \le 0, \quad i \in \bar{M}_k \tag{3'}$$

are satisfied by all $x \in P$. For each $j \in \bigcup_{i \in \bar{M}_k} Q_i$, define

$$Q(j) = \bigcup_{\substack{h \in \bar{M}_k \\ j \in Q_h}} Q_h \setminus \{j\}$$

and for $i \in \bar{M}_k$, let

$$T_i = \{ j \in Q_i \mid Q(j) \supseteq Q_h \text{ for some } h \in \bar{M}_k \}.$$

Then the inequalities

$$x_k - \sum_{j \in N_k \setminus T_k} x_j \le 0 \tag{8}$$

are satisfied by all $x \in P$.

Proof. From the definition of the sets $Q(j)$, $x \in P$ with $x_j = 1$ implies

$$\sum_{i \in Q(j)} x_i = 0$$

for all $j \in Q_i$, $i \in \bar{M}_k$. Therefore, if $j \in T_k$ then $x \in P$ with $x_j = 1$ implies

$$\sum_{i \in Q_h} x_i = 0$$

for some $h \in \bar{M}_k$, which implies $x_k = 0$, since (3') holds for $i = h$.

Hence $x \in P$ and $x_k = 1$ implies $x_j = 0$, $\forall j \in T_k$. Therefore, if the system (3') is satisfied by all $x \in P$, then so is the system (8). □

Proposition 3.1 can be used to strengthen the inequalities (2) by replacing the sets N_{ki} with $Q_i = N_{ki} \setminus T_i$. It can then again be applied to the strengthened inequalities, and so on, until no further strengthening is possible on the basis of this proposition alone.

Applying the proposition to an inequality of the system (3') consists of identifying the set T_i. This can be done by bit manipulation and the use of logical "and" and logical "or". The number of operations required is bounded by $|Q_i| \times |M_k|$.

Example 3.1. Consider again the set partitioning polytope defined by the matrix of Example 2.1, and let us use Theorem 3.1 to strengthen the inequality

$$x_1 - x_3 - x_{12} \le 0$$

associated with N_{11}. For $k = 1$, $\bar{M}_1 = \{3, 4, 5\}$, and $N_{13} = \{3, 12\}$, $N_{14} = \{3, 4\}$ and $N_{15} = \{3, 4, 5, 12\}$. Setting $Q_h = N_{1h}$, $h = 3, 4, 5$, we have

$$Q(3) = \{4, 5, 12\}, \qquad Q(12) = \{3, 4, 5\},$$

and we find that $Q(12) \supset Q_3$. Hence $T_1 = \{12\}$, and the above inequality can be replaced by

$$x_1 - x_3 \le 0.$$

Since $\{3\}$ is contained in each of N_{14} and N_{15}, the inequalities associated with these two sets can both be replaced by $x_1 - x_3 \le 0$.

A second application of Proposition 3.1 brings no further improvement.

For $k = 2$, $\bar{M}_2 = \{1, 5\}$, $N_{21} = \{13, 14\}$, $N_{25} = \{5, 15\}$ and none of the two corresponding inequalities can be strengthened via Proposition 3.1.

For $k = 5$, $\bar{M}_5 = \{1, 2, 3, 4\}$; $N_{51} = \{1, 6, 8, 9, 14\}$, $N_{52} = \{1, 2, 11, 15\}$, $N_{53} = \{2, 6, 8\}$, $N_{54} = \{2, 8, 9, 11\}$. Using Proposition 3.1 to strengthen the inequality

$$x_5 - x_1 - x_2 - x_6 - x_9 - x_{14} \le 0$$

associated with N_{15}, we set $Q_k = N_k$, $k = 1, 2, 3, 4$,

$$Q(1) = \{2, 6, 8, 9, 11, 14, 15\}, \quad Q(6) = \{1, 2, 8, 9, 14\}, \quad Q(8) = \{1, 2, 6, 9, 11, 14\},$$

$$Q(9) = \{1, 2, 6, 8, 11, 14\}, \quad Q(14) = \{1, 6, 8, 9\},$$

and we find that

$$Q(1) \supseteq Q_1, \qquad Q(9) \supseteq Q_2,$$

and hence $T_5 = \{1, 9\}$ and the inequality associated with N_{15} can be replaced by

$$x_5 - x_2 - x_6 - x_{14} \le 0.$$

When Proposition 3.1 is used to strengthen all rather than just one of the elementary inequalities in which a certain variable x_k has a positive coefficient, it is convenient to work with the set

$$Q_k = \bigcup_{i \in \bar{M}_k} Q_{ik}$$

and instead of forming the sets T_i by looking at each $j \in Q_{ik}$, $i \in \bar{M}_k$, form directly the set

$$T = \bigcup_{i \in \bar{M}_k} T_i$$

$$= \{j \in Q_k \mid Q(j) \supseteq Q_h \text{ for some } h \in \bar{M}_k\}$$

by looking once at each $j \in Q_k$, and then use $Q_{ik} \setminus T$ in place of $Q_{ik} \setminus T_i$ in (8).

The number of operations required for applying Proposition 3.1 once to all elementary inequalities in which x_k has a positive coefficient is then bounded by $|Q_k| \times |\bar{M}_k|$.

Next we discuss a second procedure for strengthening elementary inequalities.

Proposition 3.2. *Let the index sets $Q_{ik} \subseteq N_{ik}$, $i \in \bar{M}_k$, $k \in N$, be such that the inequalities*

$$x_k - \sum_{j \in Q_{ik}} x_j \le 0, \quad i \in \bar{M}_k, \quad k \in N \tag{3'}$$

are satisfied by all $x \in P$. For $i \in \bar{M}_k$, $k \in N$, define

$$U_{ik} = \{j \in Q_{ik} \mid Q_{hk} \cap Q_{ij} = \emptyset \text{ for some } h \in \bar{M}_k \cap \bar{M}_j\}.$$

Then the inequalities

$$x_k - \sum_{j \in Q_{ik} \setminus U_{ik}} x_j \le 0, \quad i \in \bar{M}_k, \quad k \in N \tag{9}$$

are satisfied by all $x \in P$.

Proof. Let $k \in N$, $i \in \bar{M}_k$ and $j \in Q_{ik}$. Then there exists $h \in \bar{M}_k \cap M_j$ such that $Q_{hk} \cap Q_{ik} = \emptyset$ and therefore adding the two elementary inequalities corresponding to Q_{ik} and Q_{hk} respectively yields

$$x_k - x_j - \sum_{\ell \in Q_{hk} \cup Q_{ik}} x_\ell \le 0$$

Since $Q_{hk} \cup Q_{ik} \subset N_h$, adding equation h of $Ax = e$ to the last inequality yields

$$x_k - x_j - \sum_{\ell \in N_h \setminus (Q_{hk} \cup Q_{ik})} x_\ell \le 1.$$

But then $x_k = 1$ implies $x_j = 0$ and therefore (3') can be replaced by (9). □

If Proposition 3.2 is applied to several elementary inequalities, then repeated applications may yield additional improvements like in the case of Proposition 3.1.

Applying Proposition 3.2 to an inequality (2') consists of identifying the set U_{ik}. Again, this can be done by bit manipulation and use of logical "and" and logical "or". The number of operations required for each $j \in Q_{ik}$ is bounded by $|\bar{M}_k \cap M_j|$, hence the total number of operations is bounded by $|Q_{ik}| \times |\bar{M}_k|$, like in the case of Proposition 3.1.

Example 3.2. Consider the set partitioning polytope defined by the matrix B

$$
\begin{array}{c|cccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 1 & & & & & & & & 1 & 1 \\
2 & 1 & & & & & & & & & 1 \\
3 & & 1 & & & 1 & & 1 & & 1 & \\
4 & & 1 & & & & 1 & & 1 & & \\
5 & & & 1 & & 1 & & & 1 & & 1 \\
6 & & & & & & 1 & & & 1 & \\
7 & & & & 1 & 1 & & 1 & & & \\
\hline
\end{array}
$$

For $k = 1$, $\bar{M}_1 = \{3, 4, 5, 6, 7\}$, and

$$N_{31} = \{2, 5, 7\}, \qquad N_{41} = \{2, 6, 8\}, \qquad N_{51} = \{3, 5, 8\},$$

$$N_{61} = \{3, 4, 6\}, \qquad N_{71} = \{4, 5, 7\}.$$

An attempt to apply Proposition 3.1 fails to strengthen any of the inequalities associated with $k = 1$. On the other hand, Proposition 3.2 can be fruitfully applied to replace both N_{51} and N_{61} with smaller sets, after applying Proposition 3.1 to $k = 2$. We have $\bar{M}_2 = \{1, 2, 5, 6, 7\}$, $\bar{M}_1 \cap \bar{M}_2 = \{5, 6, 7\}$ and $N_{52} = \{3, 10\}$, $N_{62} = \{3, 4\}$, $N_{72} = \{4\}$. Applying Proposition 3.1 we find that $Q(3) \supset N_{72}$; thus $T_5 = T_6 = \{3\}$, and the sets N_{52}, N_{62} can be replaced by $N_{52} \setminus T_5 = \{10\}$ and $N_{62} \setminus T_6 = \{4\}$ respectively.

Now writing $Q_{i1} = N_{i1}$ and $Q_{i2} = N_{i2} \setminus T_i$ for $i = 5, 6, 7$, we can apply Proposition 3.2 since

$$Q_{31} \cap Q_{41} = \emptyset$$

and thus $U_{31} = U_{41} = \{2\}$. Hence N_{31} and N_{41} can be replaced by $N_{31} \setminus U_{31} = \{5, 7\}$ and $N_{41} \setminus U_{41} = \{6, 8\}$ respectively.

4. Nonhomogeneous equivalents of the elementary inequalities and a graph-theoretical interpretation

In this section we introduce two classes of nonhomogeneous canonical inequalities, which are equivalent on LP to the elementary inequalities (3). One of these two classes of inequalities lends itself to an interesting graph-theoretical interpretation.

Proposition 4.1. *For some $k \in N$ and $i \in \bar{M}_k$, let $Q \subseteq N_i$. Then $x \in \mathrm{LP}$ satisfies any one of the following inequalities if and only if it satisfies all of them:*

$$x_k - \sum_{j \in Q} x_j \leq 0, \tag{3}$$

$$x_k + \sum_{j \in N_i \setminus Q} x_j \leq 1, \tag{10}$$

$$\sum_{j \in N_h \setminus N_i} x_j + \sum_{j \in Q} x_j \geq 1, \quad h \in M_k. \tag{11}$$

Proof. Since $Q \subseteq N_i$ and $k \notin N_i$, (10) can be obtained by adding equation i of $Ax = e$ to (3). Further, $j \in N_h$ implies $a_{hj} \neq 0$, and $j \in Q$ implies $a_{hj} = 0$; therefore $Q \cap N_h = \emptyset$. From this and the fact that $k \in N_h$, each inequality (11) can be obtained by multiplying (3) with -1 and then adding to it equation h of $Ax = e$. Since any $x \in \mathrm{LP}$ satisfies $Ax = e$, it follows that any $x \in \mathrm{LP}$ that satisfies (3), also satisfies (10) and each of the inequalities (11).

Further, (3) can be obtained from (10), as well as from each of the inequalities (11), by the reverse of the above operations, therefore any $x \in \mathrm{LP}$ which satisfies (10), or any of the inequalities (11), also satisfies (3); and, in view of the preceding paragraph, it also satisfies all the other inequalities of (10), (11). □

Remark 4.1. Proposition 4.1 remains true if the condition "$x \in \mathrm{LP}$" is replaced by "x such that $Ax = e$".

Note however, that the set packing inequality (10) and the set covering inequalities (11) are equivalent to (3) only with respect to points $x \in \mathbb{R}^n$ satisfying $Ax = e$.

To illustrate this, we assume $Q \neq \emptyset$, $N_i \setminus Q \neq \emptyset$, $N_h \setminus N_i \setminus \{k\} \neq \emptyset$, $\forall h \in M_k$, and note that:

(i) x defined by $x_k = 1$, $x_j = 0$, $j \in N \setminus \{k\}$, satisfies (10) but violates (3) and (11);

(ii) x defined by $x_j = 1$, $\forall j \in N_i$ satisfies (3) and (11), but violates (10);

(iii) $x = 0$ satisfies (3) and (10), but violates (11);

(iv) x defined by $x_j = 1$, $\forall j \in \bar{N}(i)$, $x_j = 0$, $\forall j \notin N_i$, satisfies (10) and (11), but violates (3).

Though the inequalities (10) are of the set packing type, they are not in general valid for the set packing polytope $\bar{P}$ associated with P. The next proposition states when exactly they are not.

Proposition 4.2. *The inequality* (10) *cuts off some* $x \in \bar{P} \setminus P$ *if and only if* $Q \neq N_k$.

Proof. *Necessity.* If $Q = N_k$, then $a_k a_j \neq 0$, $\forall j \in N_i \setminus Q$. Also, $a_h a_j \neq 0$, $\forall h, j \in N_i \setminus Q$. Hence the columns of A indexed by $\{k\} \cup (N_i \setminus Q)$ are pairwise nonorthogonal, and therefore (10) is satisfied by all $x \in P$.

Sufficiency. If $Q \neq N_k$, then since $a_k a_j = 0$, $\forall j \in N_k \setminus Q$, any point x such that $x_k = x_h = 1$ for some $h \in N_k \setminus Q$, $x_j = 0$ otherwise, belongs to $\bar{P}$, but violates (10). $\square$

Example 4.1. The (strengthened) elementary inequality

$$x_1 + x_3 \leq 0$$

derived in Example 3.1 is equivalent (with respect to points $x \in LP$) to:

$$x_2 + x_5 - x_6 + x_7 + x_8 - x_{12} \leq 1, \qquad \text{for } i = 3,$$

$$x_1 + x_4 - x_7 + x_8 + x_9 - x_{12} \leq 1, \qquad \text{for } i = 4,$$

$$x_1 + x_4 + x_6 - x_9 + x_{11} + x_{12} - x_{13} \leq 1, \qquad \text{for } i = 5,$$

$$x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} \leq 1, \qquad \text{for } k = 1,$$

$$x_2 + x_3 + x_{10} + x_{11} + x_{12} \leq 1, \qquad \text{for } h = 2.$$

The first one of the above set packing inequalities cuts off the point $x \in \bar{P} \setminus P$ defined by $x_1 = x_{12} = 1$, $x_j = 0$, $\forall j \neq 1, 12$; the second one cuts off $x \in \bar{P} \setminus P$ defined by $x_1 = x_4 = 1$, $x_j = 0$, $j \neq 1, 4$; while the third set packing inequality cuts off both of the above points. Note that this third inequality strictly dominates the facet of $\bar{P}$ defined by

$$x_1 - x_4 + x_{11} - x_{12} \leq 1.$$

From the practical standpoint of an implicit enumeration algorithm, every solution to the set packing problem defines a partial solution to the associated set partitioning problem. In this context the above result has to do with cutting off partial solutions to the set partitioning problem and has the following implication. We say that a set $Q \subset N_k$ is minimal if no element of Q can be removed without invalidating the (valid) inequality (3). Also, a partial solution is said to be cut off if its zero completion is cut off.

Corollary 4.2.1. *Let $k \in N$. If the sets $Q_i \subseteq N_{ik}$, $i \in \bar{M}_k$ are minimal, then the inequalities (10) cut off all partial solutions of the form $x_k = x_h = 1$ with $a_k a_h = 0$, which have no feasible completion.*

Proof. Suppose the sets Q_i, $i \in \bar{M}_k$, are minimal, and let $x_k = x_h = 1$, with $a_k a_h = 0$, be a partial solution which has no feasible completion. Then $x \in \text{vert}\, P$ and $x_k = 1$ implies $x_h = 0$, and there exists $i_* \in \bar{M}_k$ such that $h \in N_{i_*k}$. Let $(10)_{i_*}$ be the inequality (10) for $i = i_*$. From Proposition 2.3, $h \in N_{i_*} \setminus Q_{i_*}$; for otherwise $(10)_{i_*}$ remains valid when Q_{i_*} is replaced by $Q_{i_*} \setminus \{h\}$, contrary to the minimality of Q_{i_*}. But then the zero completion of $x_k = x_h = 1$, (i.e., the point obtained by setting $x_j = 0$, $j \neq k, h$), violates $(10)_{i_*}$. □

Corollary 4.2.1 suggests that the inequalities (10) can be used to enhance the orthogonality tests in implicit enumeration (see [9, 14, 19]) or in an all-binary column-generating algorithm [3]. The latter possibility is currently being explored.

The set packing inequalities (10) have a well-known graph-theoretical interpretation in terms of the intersection graph of the matrix A. We first discuss this interpretation, then use it to derive a new interpretation on a different graph which incorporates more properties of the set partitioning polytope P. For background material, see [15, 16, 17, 21].

The *intersection graph* G_A of the 0–1 matrix A has a node j for each column a_j, and an edge (i, j) for each pair of columns a_i, a_j such that $a_i a_j \neq 0$. An inequality of the form

$$\sum_{j \in V} x_j \leq 1 \tag{12}$$

is valid for the set packing polytope defined on A, i.e., is satisfied by all $x \in \bar{\mathrm{P}}$, if and only if V is the node set of a complete subgraph of G_A; and (12) is a facet of $\bar{\mathrm{P}}$ if and only if V is the node set of a clique, i.e., a maximal complete subgraph, of G_A [8, 17]. Evidently, all those inequalities (10) such that $\{k\} \cup (N_i \setminus Q)$ is the node set of a complete subgraph of G_A, are satisfied by all $x \in \bar{\mathrm{P}}$; and from Proposition 4.2, these inequalities are precisely those for which $Q = N_{ik}$. The other inequalities (10), for which $Q \neq N_{ik}$ have no interpretation on G_A.

This suggests the following interpretation on a supergraph of G_A. We define $G(A)$, the *strong intersection graph* of the matrix A, to have a node for each $j \in N$, and an edge for each pair $i, j \in N$ such that there exists no $x \in \{0, 1\}^n$ satisfying $Ax = e$, with $x_i = x_j = 1$. Clearly, G_A is a subgraph of $G(A)$, since G_A has an edge for each pair $i, j \in N$ such that there exists no $x \in \{0, 1\}^n$ satisfying $Ax \leq e$, with $x_i = x_j = 1$.

An equivalent definition of $G(A)$ is as follows. We shall say that an independent node set $S \subseteq N$ of G_A defines a *feasible partition* of N, if N can be partitioned into subsets $N_1, \ldots, N_p$, such that each N_i, $i = 1, \ldots, p$, induces a complete subgraph on G_A and contains exactly one node of S. In these terms, (i, j) is an edge of $G(A)$ if and only if there exists no independent node set S of G_A containing both i and j, which defines a feasible partition of N.

The inequalities (10) can then be interpreted on the strong intersection graph $G(A)$ as follows.

Proposition 4.3. (i) The inequality

$$\sum_{j \in V} x_j \leq 1 \tag{12}$$

is satisfied by all $x \in P$ if and only if V is the node set of a complete subgraph G' of $G(A)$.

(ii) Assume that for each $j \in V$ there exists $x \in P$ such that $x_j = 1$, and that (12) is satisfied by all $x \in P$. Then the inequality (12) is maximal if and only if V is the node set of a clique of $G(A)$.

Proof. Let G' be the subgraph of $G(A)$ induced by V.

(i) If G' is complete, then for every pair $i, j \in V$, $x \in \operatorname{vert} P$ implies $x_i = 0$ or $x_j = 0$; therefore all $x \in \operatorname{vert} P$, hence all $x \in P$, satisfy (12). If G' is not complete, there exists $x \in P$ such that $x_i = x_j = 1$ for some $i, j \in V$; but such x obviously violates (12).

(ii) If G' is a complete subgraph but not a clique of $G(A)$, then its node set V is strictly contained in the node set V' of a clique of $G(A)$, and (12) remains valid when V is replaced by V'; hence (12) is not maximal. On the other hand, if the assumption of (ii) holds and (12) is valid but not maximal, then there exists $V' \subseteq N$ such that $V' \supset V$, $V' \neq V$, and (12) remains valid when V is replaced by V'. But then V' is the node set of a complete subgraph of $G(A)$; hence G', the subgraph induced by V, is not a clique. $\square$

Example 4.1. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 1 shows G_A and $G(A)$. The thin lines are the edges of both G_A and $G(A)$, while the heavy lines are those edges of $G(A)$ not in G_A: (1,6), (1,8), (1,9), (2,4), (2,5), (2,9), (3,7), (4,9), (5,8).

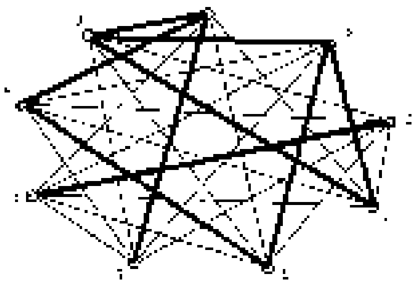


Fig. 1

If $A' = (a'_{ij})$ is the clique-node incidence matrix of $G(A)$ (with $a'_{ij} = 1$ if clique i contains node j, $a'_{ij} = 0$ otherwise), then the system $A'x \leq e'$, where $e' = (1, \ldots, 1)$, is satisfied by all $x \in P$. Furthermore, each inequality of $A'x \leq e'$ is maximal. If $\bar{P}'$ denotes the set packing polytope defined on A', i.e.,

$$\bar{P}' = \operatorname{conv}\{x \in \mathbf{R}^n \mid A'x \leq e',\ x_j = 0 \text{ or } 1,\ j \in N\},$$

we have the following obvious consequence of Proposition 4.3:

Corollary 4.3.1. *The facets of $\bar{P}'$ (which subsume or dominate those of $\bar{P}$) are valid inequalities for P.*

5. Composite inequalities of type 1

In this section we give two composition rules which can be used to combine inequalities in a certain class (which contains as a subclass the elementary inequalities of Sections 2–3) into a new inequality belonging to the same class and stronger than the sum of the inequalities from which it was obtained.

The class of inequalities to be considered, which we call composite of type 1, is that of all valid homogeneous inequalities with a single positive coefficient when stated in the form "≤ 0", and with zero coefficient in all columns j that are not orthogonal to the column k with the positive coefficient. In other words, we are referring to inequalities of the form

$$x_k - \sum_{j \in S} x_j \leq 0 \tag{13}$$

where $a_k a_j = 0$, $\forall j \in S$. The subclass of elementary inequalities is distinguished by the additional property that $S \subseteq N_i$ for some $i \in M_k$.

The first composition rule, given in the next theorem, generates a new inequality (13) from a pair of inequalities of type (13), such that the positive coefficient of the first inequality corresponds to a zero coefficient of the second one, while the positive coefficient of the second inequality corresponds to a negative coefficient of the first one.

For $k \in N$, we will denote by $L(k)$ the index set of those columns orthogonal to a_k, and by $\bar{L}(k)$ its complement; i.e.

$$L(k) = \{j \in N \mid a_j a_k = 0\}, \quad \bar{L}(k) = N \setminus L(k).$$

Proposition 5.1. *For $k, h \in N$, let $S_r \subseteq L(r)$, $r = k, h$, be such that $h \in S_k$, $k \notin S_h$, and the inequalities*

$$x_r - \sum_{j \in S_r} x_j \le 0, \quad r = k, h \tag{14}$$

are satisfied by all $x \in P$. Then all $x \in P$ satisfy the inequality

$$x_k - \sum_{j \in S} x_j \le 0 \tag{15}$$

where

$$S = (S_k \setminus \{h\}) \cup [S_h \cap L(k)].$$

Furthermore, (15) is stronger than the sum of the two inequalities (14) if and only if

$$S_k \cap [S_h \cup \bar{L}(k)] \ne \emptyset.$$

Proof. Adding the two inequalities (14) yields

$$x_k - \sum_{j \in S_k \cap S_h} 2x_j - \sum_{S'} x_j \le 0 \tag{16}$$

where

$$S' = [(S_k \setminus \{h\}) \cup S_h] \setminus S_k \cap S_h.$$

Since $x_k = 1$ implies $x_j = 0$, $\forall j \in \bar{L}(k)$, S' can be replaced in (16) by $S' \cap L(k)$. Also, since $x_k \le 1$, all coefficients 2 in (16) can be replaced by 1 without cutting off any $x \in P$. Thus (16) can be replaced by

$$x_k - \sum_{j \in S^*} x_j \le 0, \tag{16'}$$

where

$$S^* = (S_k \cap S_h) \cup [S' \cap L(k)]$$

and from the definition of S and S', we have $S^* = S$. Thus (16') is the same as (15). □

The composition rule given in Proposition 5.1 can be applied sequentially to any number of inequalities of the form (12), provided that at each step of this sequential process one can find a pair k, h of inequalities satisfying the requirement of the proposition.

Example 5.1. Consider again Example 2.1. The inequalities (3) corresponding to N_{13}, N_{24} and N_{34} respectively, after strengthening via Proposition 3.1, are

$$x_5 - x_6 - x_8 - x_{14} \le 0,$$
$$x_8 - x_{10} - x_{15} \le 0,$$

$$x_6 - x_{11} - x_{15} \le 0.$$

Using Proposition 5.1 to combine the first two inequalities, we have $k = 5$, $h = 8$, $S_5 = \{6, 8, 14\}$, $S_8 = \{10, 15\}$, $S = (\{6, 8, 14\} \setminus \{8\}) \cup \{15\}$ [since $\{10\} \notin L(5)$], and the resulting composite inequality is

$$x_5 - x_6 - x_{14} - x_{15} \le 0.$$

Since $\{10\} = S_8 \cap \bar{L}(5) \neq \emptyset$, this inequality is stronger than the sum of the inequalities from which it was obtained. Combining the new inequality with the last one of the above three inequalities, we have (the new) $k = 5$, $h = 6$, $S_5 = \{6, 14, 15\}$, $S_6 = \{11, 15\}$, $S = \{14, 15, 11\}$, and the composite inequality is

$$x_5 - x_{14} - x_{11} - x_{15} \le 0.$$

Since $\{15\} = S_5 \cap S_6 \neq \emptyset$, this inequality is again stronger than the sum of the inequalities from which it was obtained.

Next we give a second composition rule, which can be used to obtain all valid inequalities (13) for a certain index $k \in N$ from the set of elementary inequalities (3) corresponding to the same index k.

Proposition 5.2. *For some $k \in N$, let the sets $Q_i \subseteq N_k$, $i \in \bar{M}_k$, be such that the inequalities*

$$x_k - \sum_{j \in Q_i} x_j \le 0, \quad i \in \bar{M}_k \tag{3'}$$

are satisfied by all $x \in P$, and let

$$Q_0 = \bigcup_{i \in \bar{M}_k} Q_i. \tag{17}$$

Then the inequality

$$x_k - \sum_{j \in S} x_j \le 0, \tag{13}$$

where $S \subseteq N \setminus \{k\}$, is satisfied by all $x \in P$, if and only if

$$\sum_{j \in Q} a_j > c - a_k, \quad \forall Q \subseteq Q_0 \setminus S. \tag{18}$$

Proof. (i) *Necessity.* If $S \subseteq N \setminus \{k\}$ does not satisfy (18), then there exists some $Q \subseteq Q_0 \setminus S$ such that $\bar{x}$ defined by $\bar{x}_j = 1$, $j \in \{k\} \cup Q$, $\bar{x}_j = 0$ otherwise, belongs to P. But $\bar{x}$ violates (13).

(ii) *Sufficiency.* We first show that for all $x \in \text{vert}\, P$,

$$x_k = 1 \Longrightarrow x_j = 0, \quad \forall j \in N \setminus Q_k \cup \{k\}. \tag{19}$$

Since A has no zero column, $h \in L(k)$ implies that $h \in N_{ik}$ for some $i \in \bar{M}_k$. Also, $N_{ik} \subset L(k)$, $\forall i \in \bar{M}_k$. Therefore

$$\bigcup_{i \in \bar{M}_k} N_{ik} = L(k). \tag{20}$$

From Proposition 2.3 and the validity of (3') for all $x \in \operatorname{vert} P$, $x_k = 1$ implies $x_j = 0$, $\forall j \in N_{ik} \setminus Q_i$, $\forall i \in \bar{M}_k$. But

$$\bigcup_{i \in \bar{M}_k} (N_{ik} \setminus Q_i) \supseteq \bigcup_{i \in \bar{M}_k} N_{ik} \setminus \bigcup_{i \in \bar{M}_k} Q_i$$

$$= L(k) \setminus Q_k,$$

where the last equality follows from (20) and (17). Hence $x_k = 1$ implies $x_j = 0$, $\forall j \in L(k) \setminus Q_k$. Also, obviously $x_k = 1$ implies $x_j = 0$, $\forall j \in N \setminus L(k) \cup \{k\}$. But

$$[L(k) \setminus Q_k] \cup [N \setminus L(k) \cup \{k\}] = N \setminus Q_k \cup \{k\},$$

which proves (19).

Now suppose $\bar{x} \in \operatorname{vert} P$ violates (13). Then $\bar{x}_k = 1$ and $\bar{x}_j = 0$, $\forall j \in S$. Also, from (19), $\bar{x}_j = 0$, $\forall j \in N \setminus Q_k \cup \{k\}$. Hence

$$A\bar{x} = a_k + \sum_{j \in Q_k \setminus S} a_j \bar{x}_j \geq e,$$

which contradicts the condition (18) on S. $\square$

Evidently, a necessary condition for a valid inequality (13) to be maximal, is that the set S is minimal, i.e., (18) ceases to hold if S is replaced by any of its proper subsets.

Example 5.2. In Example 5.1, the inequality $x_5 - x_{11} - x_{14} - x_{15} \leq 0$ was obtained via the composition rule of Proposition 5.1. To obtain the same inequality via the rule of Proposition 5.2, let $k = 5$. Then $\bar{M}_5 = \{1, 2, 3, 4\}$, $N_{15} = \{1, 6, 8, 9, 14\}$, $N_{25} = \{1, 2, 11, 15\}$, $N_{35} = \{2, 6, 8\}$, and $N_{45} = \{2, 5, 9, 11\}$. Applying Proposition 5.2 with $Q_i = N_{i5}$, $i = 1, 2, 3, 4$, we find that

$$Q_5 = \{1, 2, 6, 8, 9, 11, 14, 15\},$$

and condition (18) is satisfied, for instance, for $S = \{11, 14, 15\}$. Hence the inequality

$$x_5 - x_{11} - x_{14} - x_{15} \leq 0$$

is satisfied by all $x \in P$.

In Proposition 5.2, the condition on S is stated in terms of a set Q_k which is the union of the sets Q_i, $i \in \bar{M}_k$, associated with the elementary inequalities (3'). The sets Q_i, and hence Q_k, are not uniquely defined, in that any $Q_i \subseteq N_{ik}$, $i \in \bar{M}_k$, for which the inequalities (17) are valid, can be used; and the smaller the set Q_0, the easier it is to generate all subsets $Q \subseteq Q_0$ which satisfy (19) and hence yield valid cuts. However, from a different perspective, setting $Q_i = N_{ik}$ for all $i \in \bar{M}_k$ gives a particularly simple expression for the family of valid cuts of the form (13).

Corollary 5.2.1. *The inequality*

$$x_k - \sum_{j \in S} x_j \geq 0 \tag{13}$$

where $S \subseteq N \setminus \{k\}$, is satisfied by all $x \in P$ if and only if

$$\sum_{j \in Q} a_j \neq e - a_k, \quad \forall Q \subseteq L(k) \setminus S. \tag{18'}$$

Proof. In Proposition 5.2, set $Q_i = N_{ik}$, $\forall i \in M_k$. Then $Q_0 = L(k)$, and the Corollary follows. □

Since the composite inequalities (13) do not have the property of elementary inequalities that $S \subseteq N_i$ for some $i \in \bar{M}_k$, there need not exist for each inequality (13) a set packing inequality that is equivalent to it on LP. On the other hand, there exist several inequalities of the more general form $\pi x \leq \pi_0$, $\pi_j \geq 0$ integer, $j \in N$, $\pi_0 > 0$ integer, which are equivalent to (13) on LP, and can be obtained by adding to (13) π_0 equations of $Ax = e$. Also, whenever $S \subseteq L(k)$, there exist several set covering inequalities which are equivalent to (13) on LP, and which can be obtained by subtracting from (13) any equation of $Ax = e$ in which x_k has a positive coefficient, and multiplying by -1 the resulting inequality.

Note also that the two strengthening procedures of Section 3 are in general not applicable to the composite inequalities (13), since both procedures are based on proofs which use the fact that for any elementary inequality $Q_i \subseteq N_{ik}$, for some $k \in N$.

Composite inequalities if type 1 can conveniently be used, along with the elementary inequalities, in a primal all-integer cutting plane algorithm for solving set partitioning problems. As we will show below, given any basic feasible integer solution, and any nonbasic variable x_k which cannot be pivoted into the basis with a value of 1 without losing feasibility, it is always possible to generate either an elementary or a composite inequality which can be used as a primal all-integer cutting plane to pivot x_k into the basis with a value of 0. These cuts are usually considerably stronger than the corresponding all-integer Gomory cuts [11] used by Young [22] and Glover [10] in their primal cutting plane algorithms, since they are derived from the special structure of the set partitioning problem. No direct comparison is available at this time with the fractional cuts proposed in [13] (see also [14]) which also use the set partitioning structure, but the cuts discussed here are obtainable directly from the matrix A, whereas those of [13] require at least partial knowledge of a fractional simplex tableau.

Finally, we note that the number of elementary inequalities is bounded by $\sum_{k\in N}|\bar{M}(k)|$, while the number of composite inequalities of type I is bounded by $\sum_{k\in N} 2^{|Q_k|}$. The latter is of course a very weak bound for the number of nonredundant inequalities of type (18), since the number of minimal sets $S \subseteq Q_k$ satisfying (18) is much less than $2^{|Q_k|}$.

To state the specific property mentioned above, let $\bar{x}$ be an integer solution to the system $Ax = e$, $x \geq 0$, with associated basis B, let I and J be the basic and nonbasic index sets respectively, and let $\bar{a}_j = B^{-1}a_j$, $\bar{a}_0 = B^{-1}e$. Suppose now that for some $k \in J$,

$$\min_{i\in I}\left\{\frac{\bar{a}_{i0}}{\bar{a}_{ik}} \,\middle|\, \bar{a}_{ik} > 0\right\} < 1,$$

i.e., x_k cannot be pivoted into the basis with value 1 without making the solution infeasible (i.e., negative in some component).

Further, for $i \in \bar{M}_k$, let $Q_i \subset N_k$ be such that the inequalities

$$x_k - \sum_{j\in Q_i} x_j \leq 0, \quad i \in \bar{M}_k, \tag{3'}$$

are valid. If there exists $i \in \bar{M}_k$ such that $Q_i \subseteq J$, then the corresponding inequality (3') can be appended to the simplex tableau and x_k can be pivoted into the basis in the row corresponding to (3') with a value of 0. Furthermore, if the reduced cost associate with column k was negative before the pivot, it will be positive after the pivot. Note also that if without the inequality (3'), x_k could have been pivoted into the basis with a fractional value, then (3') cuts off the fractional vertex of LP obtained in this way.

When $Q_i \not\subseteq J$, $\forall i \in \bar{M}_k$, i.e., when at least one of the variables x_j, $j \in Q_i$, is basic for each of the inequalities (3'), this cannot be done in general. However, in that case one can use Proposition 5.2 to generate another primal all-integer cut as follows.

Corollary 5.2.2. *Given an integer solution to the system $Ax = e$, $x \geq 0$, and an associated basis B, let I and J be the index sets for the basic and nonbasic variables respectively, and let $\bar{a}_j = B^{-1}a_j$, $j \in J$, $\bar{a}_0 = B^{-1}e$. Suppose that for some $k \in J$,*

$$\min_{i\in I}\left\{\frac{\bar{a}_{i0}}{\bar{a}_{ik}} \,\middle|\, \bar{a}_{ik} > 0\right\} < 1, \tag{21}$$

and the inequalities (3') are valid. Then

$$x_k - \sum_{j\in Q_k\cap J} x_j \leq 0, \tag{22}$$

where Q_k is defined by (17), is a valid inequality.

Proof. In view of (21), we have

$$\sum_{j\in Q} a_j \neq e - a_k, \quad \forall Q \subseteq L.$$

On the other hand, denoting $S = Q_0 \cap J = Q_0 \setminus L$, condition (18) of Proposition 5.2 becomes for this case

$$\sum_{j\in Q} a_j \neq e - a_k, \quad \forall Q \subseteq Q_0 \setminus S = L \cap Q_0.$$

Hence the condition of Proposition 5.2 is satisfied and (22) is a valid inequality. □

Thus, when no inequality (3') is available as a primal cut, the inequality (22) can serve the same purpose. A pivot in the row corresponding to (22) (and column k) has the same consequences discussed above for a pivot in the row corresponding to an inequality (3').

Example 5.3. Consider again the example of [5], whose constraint set was given in Example 2.1 and whose cost vector is

$$c = (5, 4, 3, 2, 2, 2, 3, 1, 2, 2, 1, 1, 1, 0, 0).$$

Performing all-integer primal simplex pivots produces Table 1, in which no variable with a negative reduced cost can be pivoted into the basis with a value of 1, without making the solution infeasible.

	1	$-x_1$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	x_{14}	$-x_{15}$
x_0	3	5	1	0	2	0	1	[illegible]	5	2	0
x_1	1	1	0	0	0	0	0	1	1	0	1
x_{13}	0	−1	0	0	1	−1	−1	0	1	−1	0
[illegible]	1	[illegible]	[illegible]	1	0	1	1	0	0	1	0
[illegible]	0	−1	−1	1	1	0	[illegible]	1	1	1	0
[illegible]	0	1	1	[illegible]	1	1	0	−1	−1	2	−1

Table 1.

To pivot x_{13} into the basis, one could generate from the last row the primal all integer Gomory cut

$$x_{12} - x_2 - x_3 - x_{11} - x_{15} \leq 0.$$

However, the elementary inequality

$$x_{14} - x_7 \leq 0$$

is considerably stronger. Appending it to the tableau and pivoting x_{14} into the basis produces Table 2, where s_1 stands for the slack variable associated with the cut.

	1	[illegible]	[illegible]	[illegible]	[illegible]	−[illegible]	−[illegible]	−[illegible]	−[illegible]	−[illegible]	−[illegible]
[illegible]	−3	3	−2	0	2	0	3	1	5	−2	0
x_{10}	1	1	0	0	0	0	0	1	1	0	[illegible]
x_{15}	0	−1	−1	0	1	−1	1	0	1	−1	0
x_5	1	1	2	1	0	1	1	0	0	1	0
x_4	0	1	2	1	1	0	1	1	1	1	0
x_2	0	1	4	2	−1	1	−2	−2	−3	2	−1
x_{11}	0	0	−1	0	0	0	1	0	0	0	0

Table 2

To pivot x_6 into the basis, we generate the sets N_{i6}, $i = 2, 4, 5$, and find that each of them contains at least one basic index: $N_{26} = \{10, 11, 15\}$, $N_{46} = \{4, 11\}$, $N_{56} = \{4, 5, 10\}$. Hence we form the set $Q_6 \cap J = \{4, 5, 10, 11, 15\}$, and generate a composite inequality of type 1, with $S = Q \cap J = \{11, 15\}$:

$$x_6 - x_{11} - x_{15} \le 0.$$

Note that the corresponding Gomory cut is

$$x_6 - x_2 - x_3 - x_{11} - x_{15} - x_5 \le 0.$$

Note also that the inequality $x_6 - x_{11} - x_{15} \le 0$ happens to be an elementary inequality, since $\{11, 15\} \subset \{10, 11, 15\}$, which can be obtained by applying the strengthening procedure of Proposition 3.1 to the elementary inequality $x_6 - x_{10} - x_{11} - x_{15} \le 0$.

In the next section we discuss additional options for generating primal all-integer cuts which are often stronger than or otherwise preferable to, the inequalities used here.

6. Composite inequalities of type 2

The two composition rules of Section 5 combine homogeneous canonical inequalities with a single positive coefficient (when stated in the form $\le$), and satisfying certain conditions, into a new inequality of the same form. In this section we give a composition rule which combines homogeneous canonical inequalities with (possibly) several positive coefficients into a new inequality of the same form.

Proposition 6.1. *Let K_i, $i \in T$, be pairwise disjoint subsets of N, such that $a_k a_h \neq 0$, $\forall k, h \in K$, where*

$$K = \bigcup_{i \in T} K_i;$$

and let $S(K_i) \subseteq N$, $i \in T$, be such that the inequalities

$$\sum_{j \in K_i} x_j - \sum_{j \in S(K_i)} x_j \le 0, \quad i \in T \tag{23}$$

are satisfied by all $x \in P$, and $K \cap S(K) = \emptyset$, where

$$S(K) = \bigcup_{i \in T} S(K_i).$$

Then the inequality

$$\sum_{j \in K} x_j - \sum_{j \in S(K)} x_j \leq 0 \tag{24}$$

is satisfied by all $x \in P$.

Furthermore, (24) is stronger than the sum of the inequalities (23) if and only if the sets $S(K_i)$, $i \in T$, are not all pairwise disjoint.

Proof. Any $\bar{x} \in$ vert P which violates (24) is of the form $\bar{x}_k = 1$ for some $k \in K$, $\bar{x}_j = 0$, $\forall j \in (K - \{k\}) \cup S(K)$, $\bar{x}_j$ arbitrary otherwise; but then $\bar{x}$ violates the ith inequality (23), i being the index for which $k \in K_i$. Hence (24) is satisfied by all $x \in P$.

Adding the inequalities (23) yields

$$\sum_{j \in K} x_j - \sum_{j \in S(K)} \beta_j x_j \leq 0 \tag{25}$$

where β_j is the number of sets $S(K_i)$ containing j. Clearly, (24) is stronger than (25) if and only if at least one $j \in S(K)$ is contained in more than one set $S(K_i)$, i.e., if and only if the sets $S(K_i)$ are not all pairwise disjoint. $\square$

Inequalities of the form (24) [or (23)] will be called composite of type 2. They subsume, as special cases, all the earlier inequalities discussed in this paper.

A composite inequality of type 2 always has several nonhomogeneous equivalents (on LP) of the form $\pi x \leq \pi_0$, $\pi_0 > 0$ integer, $j \in N$, $\pi_j > 0$ integer, which can be obtained by adding π_0 equalities of $Ax = e$ to (24). On the other hand, whenever $K \subseteq N_i$ for some $i \in M$, subtracting equation i of $Ax = e$ from (24) and multiplying by -1 the resulting inequality produces a set covering inequality equivalent to (24) on LP.

Since the positive coefficients of a composite inequality of type 2 are all equal to 1, and since they are computationally cheap, these inequalities can conveniently be used as primal all-integer cutting planes, along with (or instead of) and in the same way as, the elementary inequalities and the composite inequalities of type 1.

Example 6.1. In Example 5.2, after a sequence of primal integer pivots which have produced Table 1, we generated the elementary inequality

$$x_{13} - x_2 \leq 0$$

to be used as a primal all-integer cut. Pivoting in the cut-row then produced Table 2. However, the above inequality can be combined on the basis of Theorem 6.1 with two other valid elementary inequalities,

$$x_6 - x_{11} - x_{15} \leq 0$$

and

$$x_{14} - x_2 - x_{11} - x_{15} \leq 0,$$

into the composite inequality of type 2

$$x_6 + x_{13} + x_{14} - x_2 - x_{11} - x_{15} \leq 0,$$

since the vectors a_6, a_{13}, a_{14} are pairwise nonorthogonal and $\{6, 13, 14\} \cap \{2, 11, 15\} = \emptyset$.

This composite inequality has a stronger effect than the inequality $x_{13} - x_2 \leq 0$, in that it leads to optimality without additional cuts. Indeed, adding the composite cut to Table 1 and pivoting x_{11} into the basis produces Table 3:

	1	$-x_1$	[illegible]	[illegible]	$-x_3$	$-x_4$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
x_2	3	3	4	0	2	0	3	-2	2	1	-3
x_{11}	1	1	0	0	0	0	0	1	1	0	1
[illegible]	0	-1	1	0	1	-1	1	-1	0	0	-1
x_4	1	1	0	1	0	1	1	1	1	0	2
[illegible]	0	1	0	-1	1	0	1	0	0	0	1
[illegible]	0	1	-2	2	-1	1	3	1	0	-1	2
x_{13}	0	0	1	0	0	0	1	1	-1	1	-1

Table 3.

Pivoting into the basis x_{13} in place of x_5, and then x_{14} in place of x_{11} (a nondegenerate pivot) produces an optimal tableau, with $x_j = 1$, $j = 11, 12, 14$, $x_j = 0$ otherwise.

Theorem 6.1 gives a composition rule for generating inequalities of the form (24) from inequalities of the same form. However, a more general condition can be given for a composite inequality of type 2 to be valid, which makes it possible to generate inequalities of this type directly from the matrix A, rather than from other inequalities.

Proposition 6.2. *Let $K \subset N$ be such that $a_i a_j \neq 0$, $\forall i, j \in K$, and let*

$$L(K) = \{j \in N \setminus K \mid a_j a_k = 0 \text{ for some } k \in K\}.$$

Then the inequality

$$\sum_{j \in K} x_j - \sum_{j \in S} x_j \leq 0, \tag{25}$$

where $S \subseteq N \setminus K$, is satisfied by all $x \in P$ if and only if

$$\sum_{i \in Q} a_i \neq e - a_k, \quad \forall k \in K, \quad \forall Q \subseteq L(K) \setminus S. \tag{26}$$

Proof. *Necessity.* Suppose (26) is violated for some $k \in K$ and $Q \subseteq L(K) \setminus S$. Then there exists $\bar{x} \in P$ such that $\bar{x}_k = 1$ and $\bar{x}_j = 0$, $\forall j \in S$, which implies that (25) is not valid.

Sufficiency. Suppose $\bar{x} \in \operatorname{vert} P$ violates (25). Then $\bar{x}_k = 1$ for some $k \in K$ and $\bar{x}_j = 0$, $\forall j \in S$. Further, $\bar{x}_j = 0$, $\forall j \in N \setminus L(K) \cup \{k\}$. Hence

$$A\bar{x} = a_k + \sum_{j \in L(K) \setminus S} a_j \bar{x}_j = e$$

and thus $\bar{x}$ violates (26). □

Clearly, a necessary condition for the inequality (25) to be maximal, is that K be maximal and S be minimal (in the obvious sense).

Proposition 6.2 can be used to give a simple procedure for generating yet another primal all-integer cut, whenever none of several nonbasic variables can be pivoted into the basis (associated with some feasible integer solution) with a value of 1 without losing feasibility. Furthermore, generating this cut does not require knowledge of the sets Q_i or N_k.

Corollary 6.2.1. *Given an integer solution to the system $Ax = e$, $x \geq 0$, and an associated basis B, let I and J be the index sets for the basic and nonbasic variables respectively, and let $\bar{a}_j = B^{-1}a_j$, $j \in J$, $\bar{a}_0 = B^{-1}e$. Suppose that for some $h \in M$ and $K \subseteq N_h \cap J$, we have*

$$\min_{i \in I} \left\{ \frac{\bar{a}_{i0}}{\bar{a}_{ik}} : \bar{a}_{ik} > 0 \right\} < 1, \forall k \in K.$$

Then

$$\sum_{j \in K} x_j - \sum_{j \in J \setminus N_h} x_j \leq 0 \tag{28}$$

is a valid inequality.

Proof. In view of (27), we have

$$\sum_{j \in Q} a_j \neq e - a_k, \quad \forall k \in K, \forall Q \subseteq J \cap L(K) - L(K) \setminus J \cap L(K). \tag{26'}$$

Setting $S = J \cap L(K)$ and applying Proposition 6.2 produces an inequality which dominates (28), since $J \cap L(K) \subseteq J \cap N \setminus N_h$. □

Example 6.2. In Example 5.3, consider again Table 1. Choose $h \in M$ such that N_h contains as many as possible of those $j \in J$ with a negative reduced cost; i.e., let $h = 1$. Then $N_1 \cap J = \{1, 6, 7, 9, 13, 14\}$, and none of the variables x_k, $k \in N_1 \cap J$, can be pivoted into the basis with a value of 1, since each of the columns indexed by $N_1 \cap J$ has a positive coefficient in a degenerate row (in the row of x_5). Thus $K = N_1 \cap J$ is a legitimate choice, and the corresponding inequality (28) is

$$x_1 + x_6 + x_7 + x_9 + x_{13} + x_{14} - x_2 - x_3 - x_{11} - x_{12} \leq 0. \tag{28'}$$

Note that the corresponding primal all-integer Gomory cut (obtainable from the last row of Table 1) is

$$x_{16} - x_2 - x_1 - x_{11} - x_{15} \leq 0.$$

Adding to Table 1 the equation corresponding to (28) and pivoting x_{15} into the basis in place of the slack variable produces Table 4.

	1	$-x_?$	$-x_?$	$-x_?$	$-x_?$	$-x_?$	$-x_?$	$-x_{11}$	$-x_?$	$-x_?$	$-x_?$
x_1	[illegible]	6	4	3	[illegible]	3	3	2	[illegible]	[illegible]	[illegible]
x_{16}	1	[illegible]	[illegible]	0	0	0	0	[illegible]	1	0	1
$x_{1?}$	0	0	[illegible]	1	0	0	1	−1	0	0	[illegible]
$x_?$	[illegible]	0	0	0	1	0	−1	[illegible]	[illegible]	0	1
x_4	0	0	0	0	0	1	1	0	0	0	1
$x_?$	0	−2	−2	[illegible]	2	2	3	[illegible]	0	1	2
$x_{1?}$	0	[illegible]	[illegible]	1	−1	1	1	−1	−1	1	−1

Table 4.

Pivoting into the basis x_{11} in place of $x_?$, and then $x_?$ in place of x_{16} produces an optimal simplex tableau with the same solution as before ($x_j = 1$ for $j = 11, 12, 14$, $x_j = 0$ otherwise).

The material of the last two sections suggests that a primal all-integer cutting plane algorithm based on the inequalities discussed in this paper might be quite efficient. This topic is pursued in the next section.

7. A hybrid primal cutting plane/implicit enumeration algorithm

In Section 1 we defined an inequality to be valid if it is satisfied by all $x \in P$. However, in the context of solving set partitioning problems in the sense of finding an optimal solution (rather than all such solutions), it is useful to consider inequalities which are satisfied by all $x \in \text{vert}\,P$ better than some given $\bar{x} \in \text{vert}\,P$. The next theorem gives necessary and sufficient conditions for an inequality of the form (25) to be valid in this latter sense.

Proposition 7.1. *Let $\bar{x} \in \text{vert}\,P$ and let*

$$P^*(\bar{x}) = \{x \in \text{vert}\,P \mid cx < c\bar{x}\}.$$

Further, let $K \subseteq N$ be such that

$$\sum_{j \in K} x_j \geq 1, \quad \forall x \in \text{vert}\,P,$$

and let

$$I(K) = \{j \in N \setminus K \mid a_j a_k = 0 \text{ for some } k \in K\}.$$

Then the inequality

$$\sum_{j\in K} x_j - \sum_{j\in S} x_j \leq 0, \tag{29}$$

where $S \subseteq N \setminus K$, is satisfied by all $x \in \mathbf{P}^*(\bar{x})$ if and only if there exists no $x \in \mathbf{P}^*(\bar{x})$ such that $x_k = 1$ for some $k \in K$ and

$$\sum_{j\in L(K)\setminus S} a_j x_j = e - a_k. \tag{30}$$

Proof. *Necessity.* Suppose there exists $\hat{x} \in \mathbf{P}^*(\bar{x})$ such that $\hat{x}_k = 1$ for some $k \in K$, and (30) holds for $x = \hat{x}$. Then, denoting

$$W = \{k\} \cup \{L(K) \setminus S\},$$

we have

$$\sum_{j\in W} a_j \hat{x}_j = e$$

and from $\hat{x} \in \mathbf{P}$, $\hat{x}_j = 0$, $\forall j \in N \setminus W$. Hence $\hat{x}_k = 1$ and $\hat{x}_j = 0$, $\forall j \in S$, i.e. $\hat{x}$ violates (29).

Sufficiency. Suppose $\hat{x} \in \mathbf{P}^*(\bar{x})$ violates (29). Then $\hat{x}_k = 1$ for some $k \in K$ and $\hat{x}_j = 0$, $\forall j \in S$. Further, from the definition of $L(K)$, $\hat{x}_j = 0$, $\forall j \in N \setminus L(K) \cup \{k\}$. Hence

$$A\hat{x} = a_k + \sum_{j\in L(K)\setminus S} a_j \hat{x}_j = e,$$

and thus (30) holds for $\hat{x}$. $\square$

Again, a necessary condition for the inequality (29) to be maximal, is that K be maximal and S be minimal (in the obvious sense).

Proposition 7.1 can be used to generate a family of cutting planes which, in combination with implicit enumeration on a sequence of subproblems, yields a finitely converging primal algorithm for set partitioning.

Let $\bar{x}$ be a basic feasible integer solution to the linear programming relaxation of the set partitioning problem, possibly amended with some cuts of the type to be described below, and let

$$x_i = \bar{a}_{i0} - \sum_{j\in J} \bar{a}_{ij}(-t_j), \quad i \in I \cup \{0\}.$$

Denote

$$J^- = \{j \in J \mid \bar{a}_{0j} < 0\},$$

and assume that $\emptyset \neq J^- \subseteq N$ and $I \subseteq N$, i.e. no slack variable is basic or has a negative reduced cost. Assume also that

$$\min_{i\in I} \{\bar{a}_{i0}/\bar{a}_{ij} \mid \bar{a}_{ij} > 0\} < 1, \quad \forall j \in J^-.$$

i.e., pivoting into the basis any single nonbasic variable with a negative reduced cost would make the solution fractional (if $\bar{a}_{i0} > 0$) or leave it unchanged (if $\bar{a}_{i0} = 0$).

Let $i_* \in I$ be such that $J^- \cap N_{i_*} \neq \emptyset$, where, as before, $N_{i_*} = \{j \in N \mid a_{i_* j} = 1\}$, and let $j_* \in J^- \cap N_{i_*}$. Define

$$
\begin{aligned}
f_j &= \min\{0, \bar{a}_{0j} - \bar{a}_{0j_*}\}, \quad && j \in J^- \cap N_{i_*}, \\
g_j &= \min\{0, \bar{a}_{0j}\}, && j \in J \cap N \setminus N_{i_*}, \\
h_j &= \min\{0, \bar{a}_{0j} - \bar{a}_{0j_*}\}, && j \in J \cap N \setminus N_{i_*},
\end{aligned}
\tag{31}
$$

and let $J \cap N \setminus N_{i_*} = \{j(1), \ldots, j(r)\}$ be ordered so that

$$h_{j(k)} \leq h_{j(k+1)}, \quad k = 1, \ldots, r-1. \tag{32}$$

Finally, define $g_{j(0)} = 0$, and let $p \in \{0\} \cup \{1, \ldots, r\}$ be any integer such that

$$\sum_{j \in J^- \cap N_{i_*}} f_j + \sum_{j=j(0)}^{j(p)} g_j + \sum_{j=j(p+1)}^{j(r)} h_j > \sum_{j \in J^-} \bar{a}_{0j}. \tag{33}$$

Such p always exists, since (33) holds for $p = r$. The left hand side of (33) is the sum of negative reduced costs after a pivot in column j_* and the row provided by the cut (37) below.

Define

$$Q(p) = (J \setminus N_{i_*}) \cup \{j(1), \ldots, j(p)\}. \tag{34}$$

If there exists $k \in J^- \cap N_{i_*}$ and $y \in \{0, 1\}^q$, where $q = |Q(p)|$, satisfying

$$\sum_{j \in Q(p)} a_j y_j = e - a_k, \tag{35}$$

$$\sum_{j \in Q(p)} c_j y_j < c\bar{x} - c_k, \tag{36}$$

then $\hat{x} \in \mathbf{R}^n$ such that $\hat{x}_j = y_j$, $j \in Q(p)$, $\hat{x}_j = 0$, $j \in N \setminus Q(p)$, is obviously a feasible integer solution better than $\bar{x}$.

On the other hand, if this is not the case, then from Proposition 7.1 we have the following.

Corollary 7.1.1. *If there exists no $k \in J^- \cap N_{i_*}$ and $y \in \{0, 1\}^q$ satisfying (35) and (36), then the inequality*

$$\sum_{j \in J^- \cap N_{i_*}} x_j - \sum_{j=j(p+1)}^{j(r)} x_j \leq 0 \tag{37}$$

is satisfied by all $x \in P$ such that $cx < c\bar{x}$; and pivoting in row (37) and column j_* produces a simplex table with nonbasic index set $\hat{J}$ and reduced costs $\hat{a}_{0j}$ such that, denoting $\hat{J}^- = \{j \in \hat{J} \mid \hat{a}_{0j} < 0\}$,

$$\sum_{j\in J^-} |\hat{a}_{0j}| < \sum_{j\in J^-} |\bar{a}_{0j}| \tag{38}$$

and $\hat{a}_{0j} \geq 0$, $\forall j \in J \setminus N_1$ while the solution $\bar{x}$ remains unchanged.

Proof. If there exists no $k \in J^- \cap N_1$ and $y \in \{0,1\}^q$ satisfying (35) and (36), then, denoting

$$K = J^- \cap N_1 \text{ and } S = \{j(p+1), \ldots, j(r)\}, \tag{39}$$

there exists no $x \in P^I(\bar{x})$ such that $x_k = 1$ for some $k \in K$ and

$$\sum_{j\in L(K)\setminus S} a_j x_j = e - a_k. \tag{40}$$

To see this, note that

$$\begin{aligned} L(K) &\subseteq N \setminus N_1 \\ &\subseteq (I \setminus N_1) \cup (J \cap N \setminus N_1) \\ &= (I \setminus N_1) \cup \{j(1), \ldots, j(r)\} \end{aligned}$$

and hence

$$L(K) \setminus S \subseteq (I \setminus N_1) \cup \{j(1), \ldots, j(p)\} = Q(p). \tag{41}$$

Now assume that there exists $\hat{x} \in P^I(\bar{x})$ satisfying (40) for some $k \in K$. Then in view of (41), $\hat{y} \in \{0,1\}^q$ defined by $\hat{y}_i = \hat{x}_i$, $i \in L(K) \setminus S$, $\hat{y}_i = 0$ otherwise, satisfies (35), contrary to our assumption.

Thus, applying Proposition 7.1 with K and S as defined in (39), we find that the inequality (37) is satisfied by all $x \in P^I(\bar{x})$.

Further, pivoting in the row defined by (39) and column j_* produces a simplex tableau with the reduced costs

$$\hat{a}_{0j} = \begin{cases} -\bar{a}_{0j_*}, & j = j_*; \\ \bar{a}_{0j} - \bar{a}_{0j_*}, & j \in J^- \cap N_1 \setminus \{j_*\}, \\ \bar{a}_{0j} + \bar{a}_{0j_*}, & j = j(k),\ k = p+1, \ldots, r, \\ \bar{a}_{0j} & \text{otherwise,} \end{cases}$$

and since $\min\{0, \hat{a}_{0j}\} = \min\{0, \bar{a}_{0j} - \bar{a}_{0j_*}\} = 0$, the left hand side of (38) is the sum of negative reduced costs after the pivot. Thus, (33) is the same as (38). Also, $\hat{a}_{0j} \geq 0$, $\forall j \in J \setminus N_1$ since the reduced costs of t_j, $j \in J \setminus N_1$ remain unchanged. Finally, since the right hand side of (37) is zero, $\bar{x}$ also remains unchanged. □

The strength of the cut (37), as well as the computational effort involved in the search for a pair (k, y) satisfying (35), (36) (which can be carried out by implicit enumeration) depends on the size of the set $Q(p)$, viz. of the integer p. The strength of the cut increases with the size of p, but so does the computational effort involved in the search. Let $p_{\min}$ be the smallest value of p for which (33) holds. Note that when $p_{\min} = 0$, (35) cannot be satisfied for any k. Therefore in this case (37) (with $p = 0$) is always a valid cut, and there is no need for implicit enumeration to establish this fact.

Since implicit enumeration is highly efficient on small sets, but its efficiency tends to decline rapidly with the increase of the set size, a reasonable choice for p is

$$p = \max\{p_0, p_{\min}\} \tag{42}$$

where p_0 is the largest integer sufficiently small to keep the cost of the implicit enumeration acceptably low.

An algorithm based on Corollary 7.1.1 can be described as follows. Denote by J the index set of nonbasic variables, by $J \setminus N$ the index set of nonbasic slacks associated with cuts.

Step 0. Choose a value for p_0. Start with the linear programming relaxation of the set partitioning problem and go to 1.

Step 1. Perform simplex pivots which (a) leave the solution primal feasible and integer; (b) leave $a_{0j} \geq 0$, $\forall j \in J \setminus N$, and either (c) reduce the objective function value, or (d) leave the latter unchanged and reduce the absolute value of the sum of negative reduced costs. (Note that this does not exclude pivots on negative entries, or pivots which make the table fractional, provided they occur in degenerate rows. The algorithm remains valid, however, if such pivots are excluded.) When this cannot be continued, if $a_{0j} \geq 0$, $\forall j \in J$, stop: the current solution is optimal. Otherwise go to 2.

Step 2. Define i_* and j_* by

$$|J^- \cap N_{i_*}| = \max_{i \in I} |J^- \cap N_i|$$

and

$$a_{0j_*} = \min_{j \in J^- \cap N_{i_*}} |\bar{a}_{0j}|$$

respectively, order the set $J \cap N \setminus N_{i_*}$ according to (32), and choose p according to (42). Then use implicit enumeration (if necessary) to find $k \in J^- \cap N_{i_*}$ and $y \in \{0, 1\}^p$ satisfying (35), (36) (case α), or to establish that no such pair (k, y) exists (case β). Then go to step 3 (case α) or step 4 (case β).

Step 3. Pivot into the basis with a value equal to 0 each nonbasic slack variable and remove from the simplex tableau the corresponding row. Then pivot into the basis each nonbasic variable t_i such that $y_i = 1$, and go to 1.

Step 4. Generate the cutting plane (37), add it to the simplex tableau, pivot in the new row and column j_{i_*}, and go to 1.

Corollary 7.1.2. *The procedure consisting of Steps 1–4 is finite.*

Proof. Each pivot of Step 1 either decreases the objective function value z (if nondegenerate), or leaves z unchanged and decreases the absolute value σ of the sum of negative reduced costs (if degenerate). Each application of Step 2 is followed either by Step 3 or by Step 4. Step 3 consists of a sequence of pivots which decreases z. Step 4 generates a cut and performs a pivot which decreases σ, while leaving z unchanged.

In conclusion, every iteration of the algorithm either decreases z, or leaves z unchanged and decreases σ. Since σ is bounded from below by 0, z can remain unchanged only for a finite sequence of iterations. Since z is also bounded from below, the procedure is finite. □

Example 7.1. Consider again Example 5.2 (whose coefficient matrix A is given in Section 2.1). Set $p_0 = 1$ (since the example is small, we choose a small value for p_0, for otherwise no cut is needed), and apply the algorithm in its version which permits pivots only on entries equal to $+1$.

Step 1 results in Table 1 of Section 5.2.

Step 2. Define $i_* = 1$, $j_* = 14$

$$J \cap N_{i_*} N_{j_*} = \{2, 3, 11, 15\}, \quad h_2 = 0, h_3 = 0, h_{11} = -1, h_{15} = -2,$$

hence the ordering according to (32) yields

$$j(1) = 15, j(2) = 11, j(3) = 3, j(4) = 2.$$

Further, calculating the values f_i and g_{ii} from (33) we find that $p_{max} = 0$. Hence, according to (42), we choose $p = \max\{0, 1\} = 1$, and form the set

$$Q(p) = (J \setminus N_{i_*}) \cup \{j(1)\} = \{4, 5, 10, 12, 15\}.$$

Implicit enumeration shows that (35) has no solution for either $k = 13$ or $k = 14$, hence we go to

Step 4. We generate the cut

$$x_{[illegible]} - x_{[illegible]} - x_{[illegible]} - x_{[illegible]} - x_{[illegible]} \le 0,$$

and add it to the simplex tableau, with the slack denoted by s. This gives Table 5, and pivoting $x_{1[illegible]}$ into the basis in place of s yields Table 6.

	1	x_1	$-x_8$	x_3	x_5	$x_{[illegible]}$	x_{10}	x_{11}	x_{12}	x_{14}	x_{15}
x_1	-3	[illegible]	1	0	2	[illegible]	-4	[illegible]	[illegible]	-2	0
$x_{1[illegible]}$	1	1	0	0	0	0	0	[illegible]	[illegible]	0	[illegible]
$x_{[illegible]}$	0	1	0	0	1	-1	-1	0	1	-1	0
$x_{[illegible]}$	1	1	1	[illegible]	0	[illegible]	[illegible]	0	0	1	0
$x_{[illegible]}$	0	1	1	1	[illegible]	0	1	[illegible]	[illegible]	-1	0
$x_{[illegible]}$	0	1	1	2	-1	1	[illegible]	-2	[illegible]	2	1
s	0				-1		[illegible]	-1	-1	-1	

Table 5.

	1	$-x_{[illegible]}$	$-x_4$	[illegible]	$x_{[illegible]}$	$x_{[illegible]}$	$-x_{11}$	$-x_{15}$	$-x_5$	$-x_{[illegible]}$	$-x_{[illegible]}$
x_0	−3	5	1	0	0	0	−1	−1	5	2	[illegible]
x_{12}	1	1	0	0	0	0	0	1	1	0	1
x_{13}	0	−1	0	0	0	−1	0	1	0	1	0
$x_{[illegible]}$	1	1	1	1	1	1	0	[illegible]	1	−1	0
x_4	0	−1	−1	−1	[illegible]	0	0	0	0	1	0
$x_{[illegible]}$	0	1	1	2	1	1	1	0	1	2	[illegible]
$x_{[illegible]}$	0				−1		1	−2	1	1	1

Table 6

Step 6. Pivoting x_{11} into the basis in place of x_{15} and then x_{12} in place of x_5, yields the optimal solution $x_j = 1$, $j = 11, 12, 14$; $x_j = 0$ otherwise, with the following reduced costs:

	1	$-x_1$	$-x_6$	$-x_3$	$-x_2$	$-x_4$	$-x_7$	$-x_{13}$	$-x_5$	$-x_{[illegible]}$	$-x_{15}$
x_0	−2	5	0	2	1	1	1	1	3	4	0

References

[1] E. Balas, Intersection cuts from disjunctive constraints, MSRR No. 330, Carnegie-Mellon University, February 1974.

[2] E. Balas, Disjunctive programming: Cutting planes from logical conditions, in: O. Mangasarian, R.R. Meyer, and S. Robinson, eds., *Nonlinear Programming 2* (Academic Press, New York, 1975) pp. 279–312.

[3] E. Balas, R. Gerritsen and M.W. Padberg, An all-binary column generating algorithm for set partitioning. Paper presented at the ORSA/TIMS meeting in Boston, April 1974.

[4] E. Balas and R.G. Jeroslow, Canonical cuts on the unit hypercube, MSRR No. 198, Carnegie-Mellon University, August–December 1969. *SIAM J. Appl. Math.*, 23 (1972) 61–69.

[5] E. Balas and M.W. Padberg, On the set covering problem. II: An algorithm for set partitioning. *Operations Res.* 23 (1975) 74–90.

[6] E. Balas and M.W. Padberg, Set partitioning: A Survey. *SIAM Review* 18 (1976) 710–760.

[7] V. Chvátal, On certain polytopes associated with graphs. *J. Combin. Theory*, B, 18 (1975) 138–154.

[8] D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra. *Math. Programming*, 1 (1971) 168–194.

[9] R.S. Garfinkel and G.L. Nemhauser, The set partitioning problem: Set covering with equality constraints. *Operations Res.* 17 (1969) 848–856.

[10] F. Glover, A new foundation for a simplified primal integer programming algorithm. *Operations Res.*, 16 (1968) 727–740.

[11] R.E. Gomory, All integer integer programming algorithm, in: J.F. Muth and G.L. Thompson, eds., *Industrial Scheduling* (Prentice-Hall, Englewood Cliffs, NJ, 1963) pp. 193–206.

[12] R.E. Gomory and E.L. Johnson, Some continuous functions related to corner polyhedra, I and II, *Math. Programming*, 3 (1972) 23–85 and 359–389.

[13] R.G. Jeroslow, Minimal inequalities, MSRR No. 362, Carnegie-Mellon University, March 1975, revised April 1975.

[14] R.E. Marsten, An algorithm for large set partitioning problems. *Management Sci.*, 20 (1974) 774–787.

[15] G.L. Nemhauser and L.E. Trotter, Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6 (1974) 48–61.

[16] M.W. Padberg, Essays in integer programming, Ph.D. Thesis, Carnegie-Mellon University, May 1971.

[17] M.W. Padberg, On the facial structure of set packing polyhedra, *Math. Programming*, 5 (1973) 199–215.

[18] M.W. Padberg, Perfect zero-one matrices, *Math. Programming*, 6 (1974) 180–196.

[19] J.F. Pierce and J.S. Lasky, Improved combinatorial programming algorithms for a class of all zero-one integer programming problems, *Management Sci.*, 19 (1973) 528–543.

[20] J. Stoer and C. Witzgall, *Convexity and optimization in finite dimensions*, I (Springer, Berlin, 1970).

[21] L.E. Trotter, A class of facet producing graphs for vertex packing polyhedra, Technical Report No. 78, Yale University, February 1974.

[22] R.D. Young, A primal (all-integer) integer programming algorithm, *J. Res. Nat. Bur. Stds.*, 69B, 1965, pp. 213–250.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 49-64

BACKTRACKING ALGORITHMS FOR NETWORK RELIABILITY ANALYSIS*

Michael BALL** and Richard M. Van SLYKE

Network Analysis Corporation, Glen Cove, NY 11542, U.S.A.

Backtracking algorithms are applied to determine various reliability measures for networks. These algorithms are useful in analyzing the reliability of many data communication networks. We consider an undirected network where each node and each arc may be in one of two states: operative or inoperative. These states are independent random events. In addition to the more usual measure of network reliability, the probability that a specified pair of nodes can communicate, we consider more global measures such as the probability that all nodes can communicate and all operative nodes can communicate.

1. Introduction

Backtracking algorithms are very useful in solving a variety of network-related problems. They provide a framework for efficient manipulation of data with relatively small storage requirements. For example, Hopcroft [4] gives backtracking algorithms for partitioning a graph into connected components, biconnected components and simple paths; and Read [7] gives backtracking algorithms for listing cycles, paths and spanning trees of a graph. We have devised backtracking algorithms for determining certain reliability measures of a network. The algorithms are useful in analyzing the reliability of many data communications networks. For example, we have used them in the analysis of communications networks such as ARPANET in which network nodes correspond to minicomputers and network arcs correspond to transmission lines. In addition, they can be used in the analysis of radio networks in which nodes correspond to broadcast stations and arcs connect stations within broadcast range of each other.

The model we consider is an undirected network containing N_N nodes and N_A arcs. Each arc consists of an unordered pair of nodes. We do not allow self loops, that is, arcs of the form $[N, N]$. In addition we do not allow parallel arcs, that is, each arc is distinct. Each node and arc may be in either of two states: operative or inoperative. The state of a node or an arc is a random event. The state of each node and arc is independent of the state of any other node or arc. Each arc, A, and node, N, takes on the inoperative state with known probability $\mathbf{P}_A(A)$ and $\mathbf{P}_N(N)$ respectively and the operative state with probability $1-\mathbf{P}_A(A)$ and $1-\mathbf{P}_N(N)$

* This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. DAHC 15-73-C-0135.

** Now at Bell Telephone Laboratories, Murray Hill, NJ, U.S.A.

respectively. Communication can exist between a pair of nodes if they are operative and if there is a path consisting of operative nodes and arcs connecting them.

The underlying model is not new. An early reference on it is [6]. In this paper and in the majority of the work done on this problem thus far nodes are assumed to be perfectly reliable. The reliability measure most often considered is the probability that a specified pair of nodes can communicate. The reliability problems associated with many physical systems can be stated in terms of finding the probability that a specified pair of nodes can communicate in a network with perfectly reliable nodes.

We are interested in the reliability of data communications networks. For this problem we cannot assume that nodes are perfectly reliable and we require global measures of reliability. In addition to the probability that a specified pair of nodes can communicate we consider the probability that all nodes can communicate and the probability that all operative nodes can communicate. All algorithms can obtain exact answers; in addition, to allow for the analysis of larger networks we give a truncation procedure with which approximate answers can be obtained in less time.

There are basically two approaches to network reliability analysis: simulation and analytic. All known analytic methods for network reliability analysis have worst case computation time which grows exponentially in the size of the network considered. Our backtrack methods are analytic methods and are not exceptions to this trend. Hence, they are not recommended for large networks. However, results in [8] indicate that network reliability analysis is intrinsically very difficult. Simulation methods, for which computation time grows only slightly faster than linearly with network size, have been described in the literature. In our practical experience we have found that simulation techniques are suitable for large networks and are generally more flexible than analytic methods. However, they have the disadvantage that they only give approximate answers, and when a high degree of accuracy is necessary, the running time can grow quite large.

Analytic methods use basic probabilistic laws to reduce or decompose the problem. Roughly speaking these methods use some combination of enumerative and reduction techniques. Enumerative methods enumerate a set of probabilistic events which are mutually exclusive and collectively exhaustive with respect to the measure in question. Our algorithms are examples of enumerative algorithms. Reduction algorithms collapse two or more network components into one network component. The simplest example of network reduction is collapsing two series arcs into one arc.

Enumerative algorithms for finding the node pair disconnection probability with perfectly reliable nodes are given in [2,3,9]. Hansler, McAuliffe and Wilcox produce as output a polynomial in P, the constant arc failure probability. Using an APL implementation on an IBM 360–91 computer, their algorithm ran on two 9-node, 12-arc networks in a total of 18 seconds. Fratta and Montanari used a network reduction technique to reduce a 21-node, 26-arc network to an 8-node, 12-arc network. They used a FORTRAN IV implementation on an IBM–360–67 computer. Once the reduction was accomplished, they used their enumerative algorithm on the 8-node, 12-arc network to produce the exact disconnection probability. The total time for the reduction and the enumerative algorithm was 112 seconds. The reduction algorithm most probably took a small percentage of that time. Segal initially enumerates all paths in the network. He then uses the * operator ($\mathbf{P}_a^*\mathbf{P}_b = \mathbf{P}_a$ iff $a = b$) to convert the probabilities that each path operates to the probability that the node pair can communicate. This technique is especially useful when the communication paths between the node pair are restricted.

Reduction techniques have been most successful in finding the probability that a specified pair of nodes can communicate where parallel and series arcs can be collapsed into single arcs. Rosenthal [8] gives more sophisticated reduction techniques for finding other reliability measures. Rosenthal gives no computational experience; however, it appears that his techniques may be valuable for analyzing sparse networks. Generally, networks can only be reduced so far, so reduction techniques must be used in conjunction with other methods. The one exception is in the case of tree networks.

In [5] a recursive reduction algorithm is given for determining a variety of reliability measures, including all of those mentioned in this paper, on tree networks. A 500-node tree was run in 1½ seconds on a PDP-10 computer. Algorithms for general networks cannot come close to solving problems of this size.

Simulation methods have been given in [11, 12]. They provide a great deal of flexibility in the measures that can be investigated. In addition, they contain powerful sensitivity analysis capabilities. For a given number of samples, the running times increase almost linearly in the number of nodes and arcs. A 9-node, 12-arc network was run using the simulation algorithm with a FORTRAN IV implementation. The simulation algorithm produced the expected fraction of node pairs communicating and the probability that all operative nodes can communicate in 54 seconds on a PDP-10 computer.

We have implemented our algorithms using FORTRAN IV on a PDP-10 computer. The results indicate reduction in running time over the analytic algorithms listed below. In addition, our algorithms produce global reliability measures of more interest to network designers, whereas, most of the previous work was concentrated on the specified node pair problem. Our algorithms also appear to be much quicker than simulation algorithms for networks with fewer than 20 arcs. A complete summary of computational experience is given in a later section.

2. Probabilistic backtracking

Suppose we wish to enumerate all subsets of a set with a desired property. We examine elements of the set in a prescribed order. When an element is examined we decide whether or not to include it in the subset under construction. When the subset has the desired property we list it. Afterwards, we change our decision about the last element and begin adding new elements until the subset again has the desired property. If changing our decision on an element cannot produce a subset with the desired property we backup to the previous element. If this element has been considered both in and out, we backup again. If it has only been considered in one state, we change our decision on it and proceed as before. When the process terminates all subsets have been enumerated. Walker [13] has appropriately named this process "backtracking". If the enumeration is represented by a tree it can be thought of as a method for exploring a tree. More recently, it has been generalized as a method for exploring any graph and in this context it is called "depth first search" [10].

We have found this process very useful in determining the probability of a random event E. In the probabilistic context backtracking proceeds by adding probabilistic events to a stack. When the intersection of the events on the stack implies the event E, the probability of the stack configuration is added into a cumulative sum. Afterwards, we complement the top event and begin adding new events to the stack until it again implies the random event E. If complementing the top event implies that E cannot occur, we take the event off the stack and consider the new top event. If both the event and its complement have been considered, we take it off the stack. If its complement has not been considered, we complement it and proceed as before. When this process terminates, the cumulative sum will contain the probability of the event E. This is so because the events whose probabilities were added into the sum form a partition of the event E.

3. Node pair disconnection

We will first consider finding the probability that a specified node pair cannot communicate. One minus this value will give us the probability that the specified pair can communicate which is the reliability measure of interest. Henceforth, this node pair will be denoted as (S, T). For the moment, we will assume that nodes are perfectly reliable. All algorithms presented use the same basic approach. The approach is best illustrated through the specified node pair problem which is the simplest. Our algorithm embodies the general idea of [3] in a backtracking structure. Their algorithm and ours enumerate a set of "modified cut sets". A modified cutset is the assignment of one of the states, operative, inoperative or free to all arcs in the network in such a way that the inoperative arcs form a cutset with respect to the specified node pair. The probability of a modified cutset is the product of the failure probabilities of all inoperative arcs times the product of one minus the failure probabilities of all operative arcs. The modified cutsets we enumerate are mutually exclusive and collectively exhaustive with respect to the specified node pair being disconnected. Therefore, the sum of their probabilities is the probability that the specified node pair cannot communicate.

We use probabilistic backtracking to enumerate the desired set of modified cutsets. The events added to the stack are of the form "A_i inoperative" or its complement "A operative" where A is some arc. Inoperative events are added to the stack until the inoperative arcs include an $S-T$ cut. At this point the arcs on the stack will form a modified cutset and its probability will be added into a cumulative sum. The stack configuration corresponds to a modified cutset in the following manner. Arcs not included in any events on the stack are free. Other arcs are operative or inoperative depending on the type of event in which they appear. After updating the cumulative sum, the top event is changed from "A inoperative" to "A operative". The algorithm continues to proceed in the backtracking manner by again adding inoperative arcs to the stack. Two procedures are necessary to implement the algorithm. The first is a method for choosing which arcs to mark inoperative and add to the stack to form a modified cutset. In addition, after an event has been changed from "A inoperative" to "A operative" we must be able to determine if a cutset can be formed by making free arcs inoperative and adding them to the stack. If one cannot be formed we do not make A operative but simply take A off the stack. This will be the case if, when A is made operative, the operative arcs on the stack would include an $S-T$ path.

Given this basic structure, a number of algorithms could be developed depending on how the arcs to be made inoperative are chosen. Any such algorithm will fit into the following general form:

Step 0: (Initialization). Mark all arcs free; create a stack which is initially empty.

Step 1: (Generate modified cutset)

(a) Find a set of free arcs that together with all inoperative arcs will form an $S-T$ cut.

(b) Mark all the arcs found in 1(a) inoperative and add them to the stack.

(c) The stack now represents a modified cutset; add its probability into a cumulative sum.

Step 2: (Backtrack)

(a) If the stack is empty, we are done.

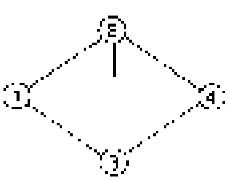
(b) Take an arc off the top of the stack.

(c) If the arc is inoperative and if when made operative, a path consisting only of operative arcs would exist between S and T, then mark it free and go to 2(a).

(d) If the arc is inoperative and the condition tested in 2(c) does not hold, then mark it operative, put it back on the stack and go to Step 1.

(e) If the arc is operative, then mark it free and go to 2(a).

Example.



$S = 1$, $T = 4$, 12 implies arc 12 is inoperative, $\overline{12}$ implies arc 12 is operative.

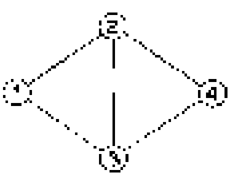
Examples of possible stack configurations:

12, 13	12, 13 are inoperative. All other arcs are free. This is a modified cutset since 12 and 13 form an $S-T$ cut and they are inoperative. If this were the stack configuration at Step 2 13 would be marked operative.
12, 13, 24, 34	12, 24, 34 are inoperative, 13 is operative. All other arcs are free. This is a modified cutset since 24 and 34 form an $S-T$ cut and they are inoperative. If this were the stack configuration at Step 2, 34 would be taken off the stack, since if it were marked operative, 13 and 34 would form an operative $S-T$ path.
12, 23, $\overline{34}$	12, 23 are inoperative; 34 is operative. All other arcs are free. This is not a modified cutset. If this were the stack configuration at Step 2 $\overline{34}$ would be removed from the stack since it is operative.

The two non-trivial operations contained in this algorithm are Step 1(a) and Step 2(c). In Step 1(a), we choose which arcs to make inoperative and put on the stack and in Step 2(c), we decide whether an inoperative arc should be complemented or whether it should be taken off the stack. Of course, the procedure used in one of these steps is closely related to the procedure used in the other.

We have devised two algorithms based on this general algorithm. Algorithm 1 enumerates a set of modified cutsets similar to the set enumerated by Hansler, McAuliffe and Wilson. Algorithm 2 enumerates a set of minimum cardinality modified cutsets with the use of a min cut algorithm.

In Algorithm 1 operative arcs form a tree rooted at node S. Inoperative arcs are adjacent to nodes in the tree. Initially, the tree consists only of node S. Node T will never be in the tree. Step 1(a) chooses all free arcs adjacent to both a node in the tree and a node not in the tree. These arcs clearly will disconnect the tree from the rest of the network and consequently, will disconnect S and T. The fact that an inoperative arc, when added to the stack, is adjacent to a node in the tree and a node not in the tree insures that, when it is marked operative, the operative arcs will continue to form a tree. In Step 2(c), an inoperative arc is taken off the stack if it is adjacent to node T.



$S = 1$, $T = 4$.

The sequence of modified cutsets generated by Algorithm 1 is:

12, 13;
12, 13, 32, 34;
12, 13, 32, 24, 34;
12, 23, 24, 13,
$\overline{12}$, 23, 24, $\overline{13}$, 34.

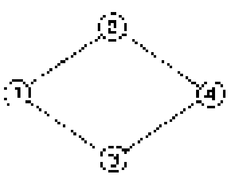
This algorithm has a very simple structure and all subprocedures take a small amount of time. The only subprocedure that cannot be done in constant time is choosing the free arcs to add to the stack, (Step 1(a)). We propose that nodes in the tree be kept on a linked list. Step 1(a) is implemented by searching the set of arcs incident to nodes on this list. This operation requires no more than $O(NA)$ time.

Theorem. *If N_a = the number of modified cutsets enumerated and N_A = the number of arcs then Algorithm 1 is $O(N_a * N_A)$.*

Proof. Any time an arc is made operative, a modified cutset is generated. As was shown earlier, in the worst case, this operation is $O(N_A)$. All operations performed in Step 2 can be done in constant time. Each operation either results in an arc being made operative and thus, a new cut being generated, or an arc being deleted from the stack. □

In Algorithm 2, operative arcs form a forest. Node S and node T are contained in different components of the forest. Step 1(a) chooses the set of free arcs of minimum cardinality that together with the inoperative arcs forms an $S-T$ cut. This minimum set of arcs is found by finding the minimum $S-T$ cut in the network with free arcs having capacity 1, inoperative arcs deleted and operative arcs having infinite capacity. The first set of free arcs added to the stack is a minimum cardinality $S-T$ cut. To implement Step 2(c), nodes in the operative tree containing S are given the label L, where L is the length of the path in the tree from the node to S. Nodes in the operative tree containing T are given the label $-L$ where L is the length of the path in the tree from the node to T. All other nodes have $L = 0$. In Step 2(c), an inoperative arc is taken off the stack if it is adjacent to nodes whose labels have opposite signs.

Example.



The sequence of modified cutsets generated by Algorithm 2 is

12, 13;
12, 13, 32, 34;
12, 13, 32, 24, 34;
$\overline{12}$, 24, 34;
$\overline{12}$, 24, $\overline{34}$, 13, 23.

Note that 1 less cutset was generated than in Algorithm 1.

Algorithm 2 enumerates an entirely different partition of the probability space than Algorithm 1. The number of events in this partition is smaller than in Algorithm 1. Algorithm 2 pays for this by the necessity of performing much more work per modified cutset generated. Again, every time an arc is made operative, the algorithm produces a modified cutset.

To find this cutset a min-cut algorithm must be performed. [2] gives a max-flow algorithm for networks with unit arc capacities that runs in $O(N_A^{3/2})$ time. This could easily be converted to a min cut algorithm suitable for our problem with the same time bound. The only other time consuming operation is the maintenance of the labels on the trees rooted at S and T. Between the generation of two modified cutsets at most one operative arc is added to the stack but as many as $N_A - 2$ may be taken off. When an operative arc is added to the stack, if it is adjacent to a node with a non-zero label, we must relabel all nodes added to the tree rooted at S or T. This requires searching the tree that has just been joined to the tree rooted at S or T. This operation requires at most $O(N_A)$ time. When an operative arc, A, is changed to free, if the nodes adjacent to it have non-zero labels, we must set the labels of the nodes that this operation disconnects from S or T to 0. We first find the node, B, adjacent to A that has the label of higher absolute value. With arc A changed to free node B will be the root of a tree not containing S or T whose nodes have non-zero labels. We search this tree and change all node labels to 0. This operation requires at most time proportional to the number of arcs adjacent to nodes whose labels were changed. Changing any set of arcs to free can change the label of each node at most once. Consequently, label changing operations between the generation of modified cutsets require at most $O(N_A)$ time.

Theorem. *If N_M = the number of modified cutsets enumerated and N_A = the number of arcs, then Algorithm 2 is $O(N_A^{3/2} N_M)$.*

Proof. The proof follows the logic in the equivalent proof for Algorithm 1 using the facts that the max-flow algorithm is $O(N_A^{3/2})$ and updating the labels is $O(N_A)$ for each modified cutset. □

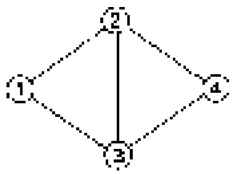
The results concerning the computational complexity of Algorithm 2 led us to believe that it would have a higher running time than Algorithm 1. Consequently, we did not code Algorithm 2 and all extensions in this paper refer to Algorithm 1. Algorithm 2 does have many interesting properties which we hope to explore later.

4. Network disconnection

A measure of the reliability of the entire network is the probability that all nodes can communicate. We chose to compute the probability that the network is disconnected which is one minus this value. Algorithm 1 extends to this case quite easily. Each modified cutset will disconnect the graph rather than only the specified node pair. (Clearly, any modified cutset which disconnects a specified node pair would also disconnect the graph.) Rather than stopping the growth of the tree when the specified node pair becomes connected, we stop it when it becomes a spanning tree. Spanning trees can easily be recognized by a count on the number of operative arcs.

In Step 2(e), we take an inoperative arc off the stack if the number of operative arcs equals $N_n - 2$.

Example.



$S = 1$.

The sequence of modified cutsets generated by Algorithm 1 with the network disconnection alteration is:

$12, 13;$
$12, \overline{13}, 32, 34;$
$12, 13, 32, 34, 42;$
$12, \overline{13}, \overline{32}, 24, 34;$
$\overline{12}, 23, 24, 13;$
$\overline{12}, 23, 24, 13, 34;$
$\overline{12}, 23, \overline{24}, 43, 13;$
$\overline{12}, \overline{23}, 24, 34.$

5. Truncation

Assuming arcs have constant failure probability, **P**, each configuration with exactly K arcs inoperative has probability $\mathbf{P}^K(1-\mathbf{P})^{N-K}$. An approximation to the node pair disconnection probability can be obtained by ignoring all network configurations with more than LIMIT arcs inoperative. If LIMIT is the smallest L such that

$$\sum_{k=0}^{L} C_A(N_A, k) \mathbf{P}^k (1 - \mathbf{P})^{N_A - k} \geq 1 - \mathrm{TOL},$$

where $C_A(N_A, k) = N_A$ things taken k at a time, then the approximation will be within TOL of the true value.

Given LIMIT, we implement this truncation procedure in our backtracking algorithm by keeping a count on the number of inoperative arcs. Whenever the addition of an arc to the stack in Step 1(i) would make the count exceed Limit, the algorithm immediately backtracks (goes to Step 2).

6. Node failures

While computations are simpler when only arcs can fail, in reality nodes are also unreliable. When considering the possibility of node failures a question arises as to the definition of network disconnection. The most obvious definition would be, "the network is disconnected any time at least one node cannot communicate with some other node" (ND1). By this definition, a network would be disconnected any time at least one node failed. An alternative definition which is much more useful for the network designer who has no control over node failure rates is, "The network is disconnected any time an operative node cannot communicate with another operative node" (ND2). Thus, if a given node is inoperative its ability to communicate with the rest of the graph is irrelevant.

(A) *Probability* {ND1}. We will consider ND1 first simply because it is easier to handle. In fact, it reduces to the problem with perfectly reliable nodes.

Let:

ANO = {all nodes operative},
NANO = {not all nodes operative}.

Then since {ANO} and {NANO} are mutually exclusive, collectively exhaustive events the law of total probability gives us:

$$\mathbf{P}\{\mathrm{ND1}\} = \mathbf{P}\{\mathrm{ND1} \mid \mathrm{ANO}\} * \mathbf{P}\{\mathrm{ANO}\} + \mathbf{P}\{\mathrm{ND1} \mid \mathrm{NANO}\} * \mathbf{P}\{\mathrm{NANO}\}.$$

$\mathbf{P}\{\mathrm{ND1} \mid \mathrm{ANO}\}$ can be found using Algorithm 1 with the network disconnection option

$$\mathbf{P}\{\mathrm{ANO}\} = \prod_{k=1}^{N_V} (1 - \mathbf{P}_N(N_k))$$

$$\mathbf{P}\{\mathrm{ND1} \mid \mathrm{NANO}\} = 1$$

$$\mathbf{P}\{\mathrm{NANO}\} = 1 - \mathbf{P}\{\mathrm{ANO}\}.$$

Thus, with one extra straight forward calculation the graph disconnection problem with node failures reduces to the graph disconnection problem with perfectly reliable nodes.

(B) *Probability* {*S*, *T* *cannot communicate*}. The definition of ND2 presents a much more difficult problem for which major modifications to the algorithm are required. First, we will again consider the node pair disconnection problem.

Let:

$S \to T$ imply S can communicate with T.

$S \not\to T$ imply S cannot communicate with T.

Then

$$\mathbf{P}\{S \not\to T\} = \mathbf{P}\{S \not\to T \mid S \text{ inop}\}^{*}\mathbf{P}\{S \text{ inop}\}$$

$$+ \mathbf{P}\{S \not\to T \mid S \text{ op}, T \text{ inop}\}^{*}\mathbf{P}\{S \text{ op}, T \text{ inop}\}$$

$$+ \mathbf{P}\{S \not\to T \mid S \text{ op}, T \text{ op}\}^{*}\mathbf{P}\{S \text{ op}, T \text{ op}\},$$

$$\mathbf{P}\{S \not\to T \mid S \text{ inop}\} = \mathbf{P}\{S \not\to T \mid S \text{ op}, T \text{ inop}\} = 1,$$

$$\mathbf{P}\{S \text{ inop}\} = \mathbf{P}_N(S),$$

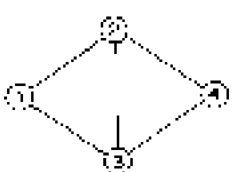
$$\mathbf{P}\{S \text{ op}, T \text{ inop}\} = (1 - \mathbf{P}_N(S))^{*}\mathbf{P}_N(T),$$

$$\mathbf{P}\{S \text{ op}, T \text{ op}\} = (1 - \mathbf{P}_N(S))^{*}(1 - \mathbf{P}_N(T)).$$

The new version of the algorithm will compute $\mathbf{P}\{S \not\to T \mid S \text{ op}, T \text{ op}\}$, i.e., we assume S and T are perfectly reliable and then find the probability that they cannot communicate.

The problem now has been reduced to enumerating a mutually exclusive, collectively exhaustive set of modified cutsets between S and T where nodes other than S and T can also "take part" in cuts. The most straightforward modification to Algorithm 1 that would compute the desired probability would be to put nodes as well as arcs on the stack. Nodes are now marked either operative, inoperative, or free. Every time an arc is made operative, the new node added to the tree is placed on the stack and marked inoperative. To disconnect this tree from the rest of the network, all free arcs between operative nodes in the tree and free nodes are added to the stack and marked inoperative. When an inoperative node is encountered in a backtrack, it is switched to operative and a new modified cutset is found in the same manner.

Consider the following example:



$S = 1$, $T = 4$. The sequence of modified cutsets generated by the suggested algorithm for the 1,4 node pair disconnection probability is:

12, 13;
12, 13, 3;
12, 13, $\bar{3}$, 32, 34;
12, $\overline{13}$, 3, $\overline{32}$, 2, 34;
12, 13, $\bar{3}$, $\overline{32}$, $\bar{2}$, 24, 34;
12, 2, 13;
$\overline{12}$, $\bar{2}$, 13, 3;
$\overline{12}$, 2, 13, $\bar{3}$, 34;
12, $\bar{2}$, 23, 24, 13;
12, $\bar{2}$, 23, 24, 13, 3;
12, $\bar{2}$, 23, 24, $\overline{13}$, $\bar{3}$, 34;
12, $\bar{2}$, $\overline{23}$, 3, 34;
12, $\bar{2}$, $\overline{23}$, $\bar{3}$, 24, 34;

where $\bar{1}$ implies node 1 is inoperative and 1 implies node 1 is operative.

A large saving can be realized by taking advantage of the equivalence between the following two events:

E_1 = node N inoperative

E_2 = node N operative; all arcs between node N and free nodes inoperative.

Notice that in the example the modified cutsets

$\{\overline{12}, 2, 13; \overline{12}, 2, \overline{13}, 3; \overline{12}, 2, \overline{13}, 3, 34\}$

are the same as

$\{12, \bar{2}, 23, 24, 13; \overline{12}, \bar{2}, 23, 24, \overline{13}, 3; \overline{12}, 2, 23, 24, 13, \bar{3}, 34\}$

except that 2 in the first set is replaced by $\bar{2}$, 23, 24 in the second set.

E_1 and E_2 are equivalent in the following sense. Given the current stack configuration the subsequent enumeration with E_1 on the stack is exactly the same as the enumeration would be with the events in E_2 on the stack. Stated probabilistically this relation is:

$$\mathbf{P}\{S \not\to T \mid E_1 \cap \text{ events on stack}\} = \mathbf{P}\{S \not\to T \mid E_2 \cap \text{ events on stack}\}.$$

If we let C_1 be the value of the cumulative sum when E_1 is placed on the stack and C_1' be the value of the cumulative sum when E_1 is changed to operative we have:

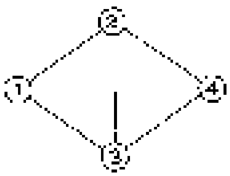
$$\mathbf{P}\{S \not\to T \mid E_1 \cap \text{ events on stack}\} = (C_1' - C_1)/\mathbf{P}\{E_1 \cap \text{ events on stack}\}.$$

This relation also applies to E_2. Thus, when we change E_1 to operative we may update the cumulative sum to C_2 to account for all the enumeration that would have proceeded with the events in E_2 on the stack where:

$$C_2 = (C_1' - C_1) * \mathbf{P}\{E_2\}/\mathbf{P}\{E_1\} + C_1'.$$

After this update, we mark the arcs in I_k inoperative, add them to the stack and immediately backtrack.

Example.



$S = 1$, $T = 4$. The sequence of modified cutsets generated for the 1, 4 disconnection probability with node failures treated implicitly is:

12, 13;
12, $\overline{13}$, 3;
cum. sum updated for 12, $\overline{13}$, $\bar{3}$, 32, 34, ...;
12, $\overline{13}$, 3, $\overline{32}$, 2, 24;
cum. sum updated for 12, 13, 3, 32, 2, 24, ...;
$\overline{12}$, 2, 13;
$\overline{12}$, 2, $\overline{13}$, 3;
cum. sum updated for 12, 2, 13, $\bar{3}$, 34, ...;
cum. sum updated for $\overline{12}$, $\bar{2}$, 23, 24, ...;
$\overline{12}$, $\bar{2}$, $\overline{23}$, 3, 24;
cum. sum updated for $\overline{12}$, $\bar{2}$, $\overline{23}$, $\bar{3}$, 34, ...

It is instructive to compare this sequence of cuts with those generated in the example for Algorithm 1.

(C) *Probability ND2.* We apply this method for treating node failures to find the probability that at least one pair of operative nodes cannot communicate (ND2). Let us first see what happens when the simple change that was made to Algorithm 1, to get the network disconnection probability, is applied to the node pair disconnect algorithm with node failures. That is, rather than changing an arc from inoperative to free, if it were incident to node T, we change it to free, if, when made operative it would complete a spanning tree of operative arcs. With this alteration, each event enumerated would disconnect node S from at least one other node in the graph. We are interested in the probability that operative nodes cannot communicate. Therefore, the probability of each event enumerated will be multiplied by the probability that at least one node not in the tree is operative. The algorithm will then produce the probability that node S does not communicate with some other operative node. In other words, the algorithm will find the probability that at least one operative node pair cannot communicate given that node S is operative. If $\{H_i\}_{i=1,\dots,N}$ are mutually exclusive and collectively exhaustive,

$$\mathbf{P}\{\mathrm{ND2}\} = \sum_{i=1}^{N_N} \mathbf{P}\{\mathrm{ND2} \mid H_i\} * \mathbf{P}\{H_i\}.$$

Let $\{S_1, S_2, \ldots, S_{N_N}\}$ be a permutation of the nodes

$$H_i = \{\text{node } S_i \text{ operative; nodes } S_1, \ldots, S_{i-1} \text{ inoperative}\}$$

Clearly $\{H_i\}_{i=1,\ldots,N_N}$ are mutually exclusive and collectively exhaustive.

$$\mathbf{P}\{H_i\} = (1 - \mathbf{P}_N(S_i)) \prod_{j=1}^{i-1} \mathbf{P}_N(S_j)$$

$\mathbf{P}\{\mathrm{ND2} \mid H_1\}$ is simply the output of the backtracking algorithm described in the preceding paragraph. $\mathbf{P}\{\mathrm{ND2} \mid H_i\}_{i>1}$ is the output of that same algorithm performed on the network with nodes $S_1, \ldots, S_{i-1}$ deleted.

Thus, the complete algorithm for computing $\mathbf{P}\{\mathrm{ND2}\}$ would compute $\mathbf{P}\{\mathrm{ND2} \mid H_1\}$ using S_1 as the root node. S_1 would then be deleted from the network and the algorithm would compute $\mathbf{P}\{\mathrm{ND2} \mid H_2\}$ using S_2 as a root node. S_2 would then be deleted and S_3 used as the root node. The algorithm would continue in this manner until all nodes had been used as root nodes. At each iteration the backtracking algorithm would be applied to a network with one less node. The formulas given above would be used to combine the output from each application of the backtracking algorithm to get $\mathbf{P}\{\mathrm{ND2}\}$.

It might appear that the running time of this algorithm would increase dramatically. This is not the case. Since the running time of the backtracking subprocedure grows exponentially, the running time on a graph with one or two nodes deleted is much less than the running time of the algorithm on the original graph. Due to this fact the running time to find $\mathbf{P}\{\mathrm{ND2}\}$ is generally less than twice the running time to find $\mathbf{P}\{\mathrm{ND1}\}$.

The choice of the ordering of the nodes is arbitrary. One good method is to pick the node with the highest degree, since on the next iteration, the maximum number of arcs will be deleted. Looking toward a possible truncation, a procedure which could be used, is to pick the node with lowest failure probability. In particular, if some node has 0 failure probability it should be chosen as S_1. In this case exactly one call to the backtracking procedure would be necessary since all probabilities subsequently enumerated must be multiplied by $\mathbf{P}_N(S_1)$.

7. Computational experience and comparison with other algorithms

We have implemented the algorithms presented in this paper to compute the probability that a specified pair of nodes can communicate and the probability that all operative nodes can communicate. The implementations consider both node and arc failures. The algorithms were coded in FORTRAN IV on a PDP-10 computer.

(A) *Probability* $\{\mathrm{ND2}\}$. A series of runs was made on networks with between 30 and 19 arcs. Table 1 is for the computation of P(ND2) with both non-zero but constant node and arc failure probabilities ($P_A(A) = 0.02$ for all A; $P_N(N) = 0.001$ for all N).

Table 1 (CPU times in seconds)

N_N	N_A	
8	10	4.05
9	13	6.45
10	15	15.49
13	17	26.21
15	19	55.10

The times include the time to read in the data.

(B) *Specified node pair.* A series of runs was made on networks with between 12 and 28 arcs. Table 2 is for the computation of the probability that a specified node pair cannot communicate. Node failures were zero and arc failures were constant ($P_N(N) = 0$ for all N, $P_A(A) = 0.02$ for all A). The networks were run with tolerances of 0.00, 0.0001 and 0.001. The true error was usually much smaller than the tolerance.

Table 2 (CPU times in seconds)

NN	NA	0.000 − TOL	0.0001 − TOL	0.001 − TOL
9	12	4.36	4.30	4.34*
10	15	7.56	4.71	3.94
11	17	8.77	5.66	5.97
15	19	17.22	9.34	6.87
19	24	21.64	16.69	17.16*
24	28	not run	not run	45.36

*In these cases there was no reduction in time between 0.0001 tolerance and 0.001 tolerance. This occurs when LIMIT has the same value for TOL = 0.0001 or TOL = 0.001.

A 10-node, 19-arc network with 0.00 tolerance ran in 26.62 seconds. Comparing this time with the time for the 15-node, 19-arc network (17.22 seconds) seems to indicate that denser networks require higher running times per number of arcs.

(C) *Comparison with other algorithms.* To compare our algorithm with the results given in [3] we added the capability of producing a failure probability polynomial in p, the constant arc failure probability. Our algorithm produced the node pair disconnection probability for the two Hansler, McAuliffe and Wilkov test networks in 11.20 seconds. As was stated in the introduction, their algorithm required 13 seconds. However, it should be noted that different computers and languages were used.

To compare our algorithm with simulation results, we computed the probability that all operative nodes can communicate on the same 9 node 12 arc network run using the simulation algorithm. The running time was 6.45 seconds. This is a marked improvement over the simulation time of 54 seconds; however, the simulation algorithm also computed the expected fraction of node pairs communicating.

References

[1] S. Even and R.E. Tarjan, Network flow and testing graph connectivity, *SIAM J. Computing* (1975) 507–518.

[2] L. Fratta and U. Montanari, A boolean algebra method for computing the terminal reliability in a communication network, *IEEE Trans. Circuit Theory*, CT-20 (1973) 203–211.

[3] E. Hansler, G. McAuliffe and R. Wilkov, Exact calculation of computer network reliability, *Networks*, 4 (1974) 95–112.

[4] J. Hopcroft and R. Tarjan, Efficient algorithms for graph manipulation, *Comm. ACM*, 16 (1973) 372–378.

[5] A. Kershenbaum and R. Van Slyke, Recursive analysis of network reliability, *Networks*, 3 (1973) 81–94.

[6] C.Y. Lee, Analysis of switching networks, *Bell System Tech. J.* 34 (1955) 1287–1315.

[7] R. Read and R. Tarjan, Bounds on backtrack algorithms for listing cycles, paths and spanning trees, *Networks*, 5 (1975) 237–252.

[8] A. Rosenthal, *Computing reliability of complex systems*, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1974.

[9] M. Segal, Traffic engineering of communications networks with a general class of routing schemes, presented at *The Fourth International Teletraffic Congress*, July, 1964.

[10] R.E. Tarjan, Depth first search and linear graph algorithms, *SIAM J. Computing*, 1 (1972) 146–160.

[11] R.M. Van Slyke and H. Frank, Network reliability analysis: Part I, *Networks* 1 (1972) 279–290.

[12] R.M. Van Slyke, H. Frank and A. Kershenbaum, Network reliability analysis: Part II, in: R.E. Barlow, ed., *Reliability and Fault Tree Analysis* (SIAM, 1975) 619–650.

[13] R.J. Walker, An enumerative technique for a class of combinatorial problems, in: Richard Bellman and Marshall Hall, Jr., eds., *Combinatorial Analysis, Proceedings Symposium on Applied Mathematics* 10 (Am. Math. Soc., Providence, RI, 1960) 91–94.

Annals of Discrete Mathematics 1 (1977) 65-78

COLORING THE EDGES OF A HYPERGRAPH AND LINEAR PROGRAMMING TECHNIQUES

Claude BERGE
University of Paris VI, Paris, France,

Ellis L. JOHNSON
IBM Research, Yorktown Heights, NY, U.S.A.

A theorem of Baranyai reduces the problem of finding the chromatic index of certain hypergraphs to a cutting stock integer programming problem. Baranyai used this result to establish the chromatic index for the complete h-uniform hypergraphs. We use a linear programming technique of Gomory and Gilmore to extend his result to two other cases: the hereditary closure of the complete h-uniform hypergraphs K_n^h for $h \leq 4$; and of the complete h-partite hypergraphs.

1. Introduction

A *hypergraph* is defined by a set X (the *vertices*) and a family $\mathscr{E} = \{E_i \mid i \in I\}$ of non-empty subsets of X (the *edges*). A *k-coloring* of the edges is a partition

$$\mathscr{E} = \mathscr{E}_1 + \mathscr{E}_2 + \cdots + \mathscr{E}_k$$

of the edge-set $\mathscr{E}$ into k classes such that all the edges in the same class are pairwise disjoint.

Let $H = (X, \mathscr{E})$ be a hypergraph. As in a graph, the *degree* $d(x)$ of a vertex $x \in X$ is the number of edges containing x. The *maximum degree* in H is denoted by

$$\Delta(H) = \max_{x \in X} d(x).$$

As in a graph, the *chromatic index* $q(H)$ is the least k for which H possesses a k-coloring of its edges. Clearly,

$$q(H) \geq \Delta(H).$$

We say that H has the *edge-coloring property* if $q(H) = \Delta(H)$. When every vertex x of H has the same degree, then H has the edge-coloring property if and only if the index set I of the edges can be partitioned into sets

$$I = I_1 + I_2 + \cdots + I_k$$

such that for each $k = 1, \ldots, k$, $\{E_i \mid i \in I_k\}$ is a partition of the nodes X.

It is not difficult to see that the determination of whether or not a given hypergraph can be k-colored, for a given k, can be expressed as determining whether or not a certain system of linear inequalities has an integer solution. The usual formulation involves a large system which is not very useful. In this paper, a theorem of Baranyai [1] will be used to relate the edge coloring problem for certain hypergraphs to the cutting stock problem, which is well-known in Operations Research (see Gilmore–Gomory [4]). The proof of Baranyai's theorem uses network flow theory and is interesting in itself. The theorem immediately gives the chromatic index of the complete h-uniform hypergraph K_n^h, which generalizes the complete graph K_n on n vertices. For other cases, it provides a much more useful linear program in order to determine $q(H)$.

In Section 3, we derive the chromatic index for the hereditary closure of the complete h-uniform hypergraph K_n^h for $h \leq 4$. The linear programming technique of Gilmore–Gomory [4] is used.

In Section 4, we investigate the edge-coloring property for the complete h-partite hypergraph $K_{n_1,n_2,\ldots,n_h}^h$ which generalizes the complete bipartite graph $K_{p,q}$.

2. The theorem of Baranyai and complete h-uniform hypergraphs

The *complete h-uniform hypergraph* K_n^h is defined by a set X of n vertices. Then, a set $E \subseteq X$ is an edge if and only if it has cardinality h. In [5], E. Lucas showed that if n is even, then the complete *graph* K_n (or K_n^2) has the edge-coloring property, i.e., $q(K_n) = \Delta(K_n)$. This result is now very well-known in Graph Theory and Statistics. Lucas also conjectured that if n is a multiple of 3, then the complete 3-uniform hypergraph K_n^3 has the edge-coloring property. This result was proven for $n = 9$ by Walecki (see Lucas [5]) and for all $n = 3k$ by R. Peltesohn [6]. Her proof was long and exhaustive. Finally, Baranyai established the chromatic index for all K_n^h [1].

In a different area, P. Gilmore and R. Gomory devised a linear programming approach to the cutting-stock problem [4]. In that problem, one assumes that a supply of rolls of paper, each roll of stock length n, is maintained. From these rolls are to be cut k_i pieces of length r_i, for $i = 1, \ldots, q$. In order to minimize the wastage, we want to determine the least number

$$g(n; k_1 \times r_1, k_2 \times r_2, \ldots, k_q \times r_q)$$

of rolls that is needed. Necessarily,

$$r_i \leq n, \quad i = 1, 2, \ldots, q.$$

Clearly, we have

$$g(n; k \times r) = \lceil k / \lfloor n/r \rfloor \rceil,$$

where $\lfloor \lambda \rfloor$ is the largest integer smaller than or equal to λ and $\lceil \lambda \rceil$ is the smallest integer larger than or equal to λ.

In discussing this problem, we will refer to the stock lengths as being *sticks* of length n rather than rolls; only the length n is important, and we prefer to think of it as a linear form.

Baranyai's Theorem. *Consider the hypergraph $K_n^{r_1} + K_n^{r_2} + \cdots + K_n^{r_p}$ on a set X of n vertices, whose edges are all the r_1-subsets, all the r_2-subsets, …, all the r_p subsets of X. (If two r_i's are equal, this hypergraph has multiple edges). Then the chromatic index $q(K_n^{r_1} + \cdots + K_n^{r_p})$ is equal to*

$$\xi\left(n; \binom{n}{r_1} \times r_1, \binom{n}{r_2} \times r_2, \ldots, \binom{n}{r_p} \times r_p\right).$$

In other words, it is possible to color the

$$\sum_{i=1}^{p} \binom{n}{r_i}$$

edges of this hypergraph with q colors if and only if it is possible to cut, from a stock of q sticks of length n, $\binom{n}{r_1}$ pieces of length r_1, $\binom{n}{r_2}$ pieces of length r_2, …, $\binom{n}{r_p}$ pieces of length r_p. Each stick corresponds to a color. It is obvious that from a coloring, one can cut the sticks as required. The importance of the theorem is the other direction, in which the result says that one need not actually determine the particular edges to be colored a given color, but instead one need only determine the coloring "pattern" corresponding to each stick.

Denote by h_{ij} the number of pieces of length r_i that we cut from the jth stick. Then the cutting stock problem is feasible if and only if $(n; r_1, \ldots, r_p)$ and the $p \times q$ matrix (h_{ij}) satisfy:

(1) $\quad r_i$ integer, $0 \le r_i \le n$, $i = 1, \ldots, p$;

(2) $\quad h_{ij}$ integer and $h_{ij} \ge 0$, $i = 1, \ldots, p$ and $j = 1, \ldots, q$;

(3) $\quad \sum_{j=1}^{q} h_{ij} = \binom{n}{r_i}$, $i = 1, \ldots, p$;

(4) $\quad \sum_{i=1}^{p} r_i h_{ij} \le n$, $j = 1, \ldots, q$.

We now turn to the proof of Baranyai's theorem. There is no essential difference between the proof here and that in [1]; we give it in full for three reasons: it is an interesting use of network flow theory; [1] may not be readily available to the reader; and the proof here is a little simpler.

Proof of the theorem. We shall show the following: if $(n; r_1, \ldots, r_p)$ and (h_{ij}) satisfy (1), (2), (3), and (4), then there exist subsets

$$E_k^{ij}, \quad k = 1, 2, \ldots, h_{ij}$$

such that:

(A) $\{E^k_{ij} \mid j = 1, \ldots, q$ and $k = 1, \ldots, h_{ij}\}$ is isomorphic to $K^{r_i}_n$, for $i = 1, \ldots, p$;

(B) $\{E^k_{ij} \mid i = 1, \ldots, p$ and $k = 1, \ldots, h_{ij}\}$ is a family of pairwise disjoint edges, corresponding to color j, for $j = 1, \ldots, q$.

The proof is by induction on n. Clearly, the result is true for $n = 1$ and $n = 2$.

To prepare for the induction, we first remove every $r_i = 0$ and $r_i = n$ and remove from (h_{ij}) the corresponding rows. It is clear that if the result is true for a given n without such r_i, then it is true with such r_i.

We next show that there exist integers x_{ij} for $i = 1, \ldots, p$ and $j = 1, \ldots, q$ such that

(5) $$\lfloor h_{ij} r_i / n \rfloor \le x_{ij} \le \lceil h_{ij} r_i / n \rceil ,$$

(6) $$\left\lfloor \sum_{j=1}^{q} h_{ij} r_i / n \right\rfloor \le \sum_{j=1}^{q} x_{ij} \le \left\lceil \sum_{j=1}^{q} h_{ij} r_i / n \right\rceil ,$$

(7) $$\left\lfloor \sum_{i=1}^{p} h_{ij} r_i / n \right\rfloor \le \sum_{i=1}^{p} x_{ij} \le \left\lceil \sum_{i=1}^{p} h_{ij} r_i / n \right\rceil .$$

This demonstration uses the fact that if a network has integer bounds on each arc flow and has a feasible flow, then it has a feasible integer flow. Specifically, construct a transportation network with source s, sink t, and two sets $P = \{1, 2, \ldots, p\}$ and $Q = \{1, 2, \ldots, q\}$ of vertices; the arcs are all the pairs (s, i) with $i \in P$, (i, j) with $i \in P$ and $j \in Q$, and (j, t) with $j \in Q$. The constraints on the flow ϕ are

$$\left\lfloor \sum_{j=1}^{q} h_{ij} r_i / n \right\rfloor \le \phi(s, i) \le \left\lceil \sum_{j=1}^{q} h_{ij} r_i / n \right\rceil$$

for every source arc (s, i), $i \in P$,

$$\lfloor h_{ij} r_i / n \rfloor \le \phi(i, j) \le \lceil h_{ij} r_i / n \rceil$$

for every intermediate arc (i, j), $i \in P$, $j \in Q$,

$$\left\lfloor \sum_{i=1}^{p} h_{ij} r_i / n \right\rfloor \le \phi(j, t) \le \left\lceil \sum_{i=1}^{p} h_{ij} r_i / n \right\rceil$$

for every sink arc (j, t), $j \in Q$.

Clearly, $\phi(s, i) = \sum_j h_{ij} r_i / n$, $\phi(i, j) = h_{ij} r_i / n$, and $\phi(j, t) = \sum_i h_{ij} r_i / n$ is a feasible flow. Hence, there exists an integer feasible flow ψ. Letting $x_{ij} = \psi(i, j)$, $i \in P$, $j \in Q$, gives (5), (6), and (7).

Now, consider the vector $(n - 1, r_1 - 1, r_2 - 1, \ldots, r_p - 1, r_1, r_2, \ldots, r_p)$ and the $(2p) \times q$ matrix (h'_{ij}), where

$$h'_{ij} = \begin{cases} x_{ij} & \text{for } i = 1, \ldots, p \text{ and } j = 1, \ldots, q; \\ h_{i-p,j} - x_{i-p,j} & \text{for } i = p + 1, \ldots, 2p \text{ and } j = 1, \ldots, q. \end{cases}$$

We next show that this new vector and new matrix satisfy the conditions (1), (2), (3), and (4), corresponding to them. Condition (1) is true by having removed $r_i = 0$ and $r_i = n$ beforehand. Condition (2) is true by e_{ij} integer and

$$0 \le \lfloor h_{ij} r_i / n \rfloor \le e_{ij} \le \lceil h_{ij} r_i / n \rceil \le h_{ij}$$

where the last inequality is by $r_i \le n$ and h_{ij} integer. To show (3), note that by (3) for the old vector and matrix,

$$\sum_{j=1}^{q} \frac{h_{ij} r_i}{n} = \frac{r_i}{n} \sum_{j=1}^{q} h_{ij} = \frac{r_i}{n} \binom{n}{r_i} = \binom{n-1}{r_i - 1}.$$

Hence, (3) now follows from (6) for $i = 1, \ldots, p$. For $i = p + 1, \ldots, 2p$, (3) follows from

$$\sum_{j=1}^{q} (h_{ij} - e_{ij}) = \sum_{j=1}^{q} h_{ij} - \sum_{j=1}^{q} e_{ij} = \binom{n}{r_i} - \binom{n-1}{r_i - 1} = \binom{n-1}{r_i}$$

by the binomial formula.

Condition (4) for the new vector and matrix is equivalent to

$$\sum_{i=1}^{p} (r_i - 1) e_{ij} + \sum_{i=1}^{p} r_i (h_{ij} - e_{ij}) \le n - 1, \quad \text{or}$$

$$\sum_{i=1}^{p} r_i h_{ij} - \sum_{i=1}^{p} e_{ij} \le n - 1.$$

From (4) for the old vector and matrix and from (7),

$$\sum_{i=1}^{p} e_{ij} = 0 \quad \text{or} \quad 1.$$

$$\sum_{i=1}^{p} r_i h_{ij} = n \implies \sum_{i=1}^{p} e_{ij} = 1.$$

Hence, the required inequality is satisfied.

Let Y be a set of $n - 1$ vertices. By the induction hypothesis, there exist subsets F_{ij}^k of Y for $k = 1, \ldots, h'_{ij}$ such that:

(C) $\{F_{ij}^k \mid j = 1, \ldots, q \text{ and } e_{ij} = 1\}$ is isomorphic to $K_{n-1}^{r_i - 1}$ for $i = 1, \ldots, p$;

(D) $\{F_{ij}^k \mid j = 1, \ldots, q \text{ and } k = 1, \ldots, h_{i-p,j} - e_{i-p,j}\}$ is isomorphic to $K_{n-1}^{r_i}$ for $i = p + 1, \ldots, 2p$;

(E) $\{F_{ij}^k \mid i = 1, \ldots, 2p \text{ and } k = 1, \ldots, h'_{ij}\}$ is a family of pairwise disjoint edges for $j = 1, \ldots, q$.

We have already seen that

$$\sum_{i=1}^{p} e_{ij} = 0 \text{ or } 1,$$

so that the family of disjoint edges

$$\{F_{ij}^k \mid i = 1, \ldots, 2p \text{ and } k = 1, \ldots, h'_{ij}\}$$

has at most one edge of cardinality $r_i - 1$; all others being of cardinality r_i. Let X be an n-set obtained from Y by adjoining one new vertex α, and let (for $k = 1, \ldots, k_i$)

$$F_{ij}^{k} = \begin{cases} F_{ij}^{k} \cup \{\alpha\} & \text{if } k - r_i = 1, \\ F_{i,j-1}^{k} & \text{otherwise,} \end{cases}$$

for each $i = 1, \ldots, p$ and $j = 1, \ldots, q$. This new family F_{ij}^{k} satisfies (A) and (B), completing the induction.

From this result follows:

Corollary 1. *The chromatic index of the complete r-uniform hypergraph on n vertices is*

$$q(K_n^r) = \left\lceil \binom{n}{r} \bigg/ \lfloor n/r \rfloor \right\rceil.$$

Corollary 2. *The hypergraph K_n^r has the edge-coloring property if and only if n is a multiple of r.*

Proof of Corollary 2. When n is a multiple of r,

$$q(K_n^r) = \left\lceil \frac{r}{n}\binom{n}{r} \right\rceil = \binom{n-1}{r-1} = \Delta(K_n^r).$$

When n is not a multiple of r, both of the rounding operations in determining $q(K_n^r)$ increase it above $\Delta(K_n^r)$, and at least one increases the expression strictly.

3. The hereditary closure of the h-uniform hypergraph

For hypergraph H on X with edges $\{E_i \mid i \in I\}$. The *hereditary closure* $\mathrm{cl}(H)$ is the hypergraph on X where S is an edge if and only if

$$\phi \neq S \subseteq E_i \quad \text{for some} \quad i \in I.$$

We define $\mathrm{cl}(K_n^h)$ to be the hereditary closure of the complete h-uniform hypergraph K_n^h. Thus, E is an edge of $\mathrm{cl}(K_n^h)$ if and only if

$$\phi \neq E \subseteq X, \quad \text{and}$$

$$|E| \leq h.$$

From Baranyai's theorem, we show

Corollary 3. *Let $A = (a_i \mid i = 1, 2, \ldots, h)$ be vector satisfying:*

$$\text{(8)} \qquad \sum_{i=1}^{h} i a_i \leq n, \quad \textit{and}$$

(9) $\qquad a_0 \geq 0$ *and* a_i *integer.*

Suppose, further, that every distinct solution to (8) *and* (9) *is represented as a column of* A. *Then*

(10) $$q(\mathrm{cl}(K_n^h)) = \min \sum_j x_j$$

$$\text{s.t. } Ax = b, \text{ and } x \geq 0 \text{ and integer},$$

where $b_i = \binom{n}{i}$, $i = 1, \ldots, h$. *Further,* $\mathrm{cl}(K_n^h)$ *has the edge coloring property if and only if there is a solution to* (10) *such that for every* $x_j \geq 1$, *the column* A^j *satisfies* (8) *with equality.*

Proof. Use

$$\mathrm{cl}(K_n^h) = K_n^1 + K_n^2 + \cdots + K_n^h$$

and Baranyai's theorem, where the matrix H has x_j copies of column A^j of A.

We now establish the chromatic index of $\mathrm{cl}(K_n^h)$ for $h \leq 4$. The result for $h = 3$ was given by Bermond [3].

Theorem 2. *The hereditary closure* $\mathrm{cl}(K_n^h)$, $h \leq 4$, *has the edge coloring property except for the following cases:*

$h = 3$, $n \equiv 1 \pmod 3$, $n \geq 7$, *then*

$$q(\mathrm{cl}(K_n^3)) = \Delta(\mathrm{cl}(K_n^3)) + \lceil \tfrac{1}{4}(n-4) \rceil ;$$

$h = 4$, $n \equiv 1 \pmod 4$, $n \geq 9$, *then*

$$q(\mathrm{cl}(K_n^4)) = \Delta(\mathrm{cl}(K_n^4)) + \lceil \tfrac{1}{6}(n-5) \rceil ,$$

$h = 4$, $n \equiv 2 \pmod 4$, $n \geq 10$, *then*

$$q(\mathrm{cl}(K_n^4)) = \Delta(\mathrm{cl}(K_n^4)) + \lceil \tfrac{1}{8}(n-7) \rceil .$$

Proof. The case $h = 1$ is trivial. For $h = 2$ and n even, the edge coloring property for $\mathrm{cl}(K_n^2)$ follows from the same property for K_n^1 and K_n^2. For $h = 2$ and n odd, we need only give the "pattern": in this case, each color is one singleton and $(n-1)/2$ 2-edges, and there are n colors. The corresponding column of A is

$$\begin{bmatrix} 1 \\ \tfrac{1}{2}(n-1) \end{bmatrix}$$

and satisfies (8) with equality. By Corollary 5, $\mathrm{cl}(K_n^2)$ has the edge coloring property.

For $h = 3$ and $n \equiv 0 \pmod 3$, the result follows from $\mathrm{cl}(K_n^3) = K_n^3 + \mathrm{cl}(K_n^2)$ and Corollary 2 applied to K_n^3, while $\mathrm{cl}(K_n^2)$ has the edge coloring property for all n.

For $h = 3$ and $n \equiv 2 \pmod 3$, the required solution to (10) has columns A^j given by

$$\begin{bmatrix} 0 \\ 1 \\ \frac{1}{2}(n-2) \end{bmatrix}, \qquad \begin{bmatrix} n \\ 0 \\ 0 \end{bmatrix}$$

with corresponding values of $x_j = \binom{n}{3}$ and $x_j = 1$.

Before treating the case $k = 3$ and $n \equiv 1 \pmod 3$, we show the edge coloring property for $k = 4$ and $n \equiv 0 \pmod 4$ or $n \equiv 3 \pmod 4$. For $n \equiv 3 \pmod 4$, the required solution to (10) has columns

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \frac{1}{4}(n-3) \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ \frac{1}{2}(n-1) \\ 0 \\ 0 \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} 0 \\ \frac{1}{2}n \\ 0 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} n \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

depending on whether n is odd or even. The corresponding values of x_j are $\binom{n}{3}$ and n, or $n-1$ and 1.

For $n \equiv 0 \pmod 4$, the solution to (10), provided $n \geq 12$, has columns

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{n}{4} \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 4 \\ \frac{n-12}{4} \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ \frac{n-1}{2} \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ \frac{n}{2} \\ 0 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} n \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

depending on whether n is odd or even. The corresponding values of x_j are

$$\frac{n(n-1)(n-2)}{8}, \; \frac{1}{4}\binom{n}{3}, \quad \text{and } n, \text{ or } n-1 \text{ and } 1.$$

For $n = 4$ and $n = 8$, the edge coloring property can be verified directly.

Three cases remain: $k = 3$, $n \equiv 1 \pmod 3$; $k = 4$, $n \equiv 1 \pmod 4$; and $k = 4$, $n \equiv 2 \pmod 4$. In each of these three cases, we will first exhibit the optimum linear programming solution associated with (10). In a minimization problem in integers, such as (10), the rounded up linear programming objective value is a clear lower bound on the integer programming objective. In each of the three cases, we shall show that an integer solution to (10) achieves that bound, thus establishing the optimal integer objective value and, thereby, $q(\mathrm{cl}(K_n^k))$.

We first treat $k = 3$, $n \equiv 1 \pmod 3$. In this case, an optimum linear programming basis is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ \frac{n-1}{3} & \frac{n-4}{3} & \frac{n-1}{3} \end{bmatrix}$$

To prove optimality, we give the primal and dual solutions corresponding to this basis and show that the dual is feasible to $\pi A \leq 1$. Optimality is then assured by the complementary slackness theorem or, alternatively, can be verified by showing equality of the two objective values.

The primal solution is

$$n, \quad \frac{n(n-1)}{4}, \quad \frac{n(n-4)}{4}$$

and the dual solution is

$$0, \quad \frac{3}{2(n-1)}, \quad \frac{3}{n-1}.$$

Clearly, the primal is non-negative for $n \geq 4$, although it may fail to be integer. To show that the dual is feasible requires showing

$$\frac{3}{2(n-1)}a_2 + \frac{3}{n-1}a_3 \leq 1$$

whenever $2a_2 + 3a_3 \leq n$, and $a_2, a_3 \geq 0$ and integer. This fact can be easily demonstrated.

The objective value is

$$n + \frac{n(n-1)}{4} + \frac{n(n-4)}{4} = \frac{n(2n-1)}{4},$$

and since

$$\Delta(\mathrm{cl}(K)^2) = 1 + \binom{n-1}{1} + \binom{n-1}{2} = \frac{n^2 - n + 2}{2},$$

the objective value is

$$\Delta(\mathrm{cl}(K)^2) + \frac{n-4}{4}.$$

It remains to show that rounding this objective value up to the nearest integer is the objective value for some integer solution to (10).

In preparation, observe that the objective of (10) is not changed if $Ax = b$ is changed to the seemingly weaker $Ax \geq b$, because for any column of A, every non-negative integer column less than that column is also a column of A. For $Ax \geq b$ one can obtain an integer solution by rounding up the linear programming answer, which here is

$$n, \quad \left\lceil\frac{n(n-1)}{4}\right\rceil, \quad \left\lceil\frac{n(n-4)}{4}\right\rceil.$$

Since n is always integer, let us write

$$\tfrac{1}{4}n(n-1) = I_2 - f_2, \; 0 \leq f_2 < 1,$$

$$\tfrac{1}{4}n(n-4) = I_3 - f_3, \; 0 \leq f_3 < 1,$$

so that the objective value is

$$n + I_2 + I_3 + f_2 + f_3 .$$

If either f_2 or f_3 (or both) is zero, then rounding the variables up gives an objective value, corresponding to an integer solution, of

$$n + I_2 + I_3 + \lceil f_2 \rceil + \lceil f_3 \rceil = \lceil n + I_2 + I_3 + f_2 + f_3 \rceil$$

which says that there is an integer solution whose objective is equal to the rounded up linear programming objective. Hence, assume that $f_2 > 0$ and $f_3 > 0$. If $f_2 + f_3 > 1$, then the same result holds. It is not possible that $f_2 + f_3 = 1$ because then the objective value

$$\Delta(\mathrm{cl}(K)_n^c) - \tfrac{1}{3}(n-4)$$

is integer. Then, so is

$$\tfrac{1}{3}n(n-4) = I_3 + f_3$$

an integer, contradicting $f_3 > 0$.

The remaining case is $f_2 + f_3 < 1$. Now f_2 must be 1/2 since

$$\tfrac{1}{2}n(n-1) - \tfrac{1}{2}\binom{n}{2} = I_2 + f_2 ,$$

so f_3 must be one-fourth. Consider the integer solution to (10):

$$n\begin{bmatrix}1\\0\\\frac{1}{3}(n-1)\end{bmatrix} + I_2\begin{bmatrix}0\\2\\\frac{1}{3}(n-4)\end{bmatrix} + I_3\begin{bmatrix}0\\0\\\frac{1}{3}(n-1)\end{bmatrix} + \begin{bmatrix}0\\1\\\frac{1}{3}(n-4)\end{bmatrix} = \begin{bmatrix}n\\2I_2+1\\\frac{1}{3}(n-1)+\frac{1}{3}I_2(n-4)+\frac{1}{3}I_3(n-1)+\frac{1}{3}(n-4)\end{bmatrix} .$$

The objective value is

$$n + I_2 + I_3 + 1 = \lceil n + I_2 + I_3 + f_2 + f_3 \rceil$$

and the solution satisfies $Ax \ge b$ provided

$$2I_2 + 1 \ge 2(I_2 + f_2) \text{ and}$$

$$n\frac{n-1}{3} + I_2\frac{n-4}{3} + I_3\frac{n-1}{3} + \frac{n-4}{3} \ge n\frac{n-1}{3} + (I_2 + f_2)\frac{n-4}{3} + (I_3 + f_3)\frac{n-1}{3} .$$

The first is clear by $f_2 = 1/2$. The second is equivalent to

$$\frac{n-4}{3} \ge f_2\frac{n-4}{3} + f_3\frac{n-1}{3} = \frac{1}{2}\frac{n-4}{3} + \frac{1}{4}\frac{n-1}{3} ,$$

$$\frac{1}{2}\,\frac{n-4}{3} \geq \frac{1}{4}\,\frac{n-1}{3},$$

or $n \geq 7$. Since $n = 4$ can be treated specifically, the result follows.

Having developed the ideas, the remaining two cases will be treated more briefly. For $k = 4$ and $n \equiv 1 \pmod 4$, an optimal linear programming basis is, for $n \geq 9$,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ \frac{1}{4}(n-1) & \frac{1}{4}(n-5) & \frac{1}{4}(n-9) & \frac{1}{4}(n-1) \end{bmatrix}$$

with primal solution

$$n, \quad \binom{n}{2}, \quad \frac{1}{9}\,\frac{n(n-1)}{2}(n-5), \quad \frac{1}{9}\,n^2(n-5),$$

which is clearly non-negative for $n \geq 5$. The dual solution is

$$0, \quad \frac{1}{3}\,\frac{4}{(n-1)}, \quad \frac{2}{3}\,\frac{4}{(n-1)}, \quad \frac{4}{(n-1)},$$

which can be shown to satisfy $\pi A \leq 1$. The objective value is

$$\Delta(\mathrm{cl}(K_n^k)) + \tfrac{1}{9}n(n-5).$$

As before, only two variables have non-integer values and can be written as

$$\tfrac{1}{9}\cdot\tfrac{1}{2}n(n-1)(n-5) = I_3 + f_3,$$

$$\tfrac{1}{9}n^2(n-5) = I_4 + f_4.$$

If only one of f_3, f_4 is positive, the result follows as before. If $f_3 + f_4 > 1$, then the result also follows as before. Also, $f_3 + f_4 = 1$ is impossible because then the objective value, and hence

$$\tfrac{1}{9}n(n-5),$$

would be integer so that $f_3 = 0$ follows. The remaining case is $f_3 + f_4 < 1$ and f_3, $f_4 > 0$. But f_3 must be $\frac{1}{3}$ or $\frac{2}{3}$ because one of n, $n-1$, $n-5$ is divisible by three. Consider first

(i) $f_3 = \frac{1}{3}$.

Then the solution

$$n\begin{bmatrix} 1 \\ 0 \\ 0 \\ \frac{1}{4}(n-1) \end{bmatrix} + \binom{n}{2}\begin{bmatrix} 0 \\ 1 \\ 1 \\ \frac{1}{4}(n-5) \end{bmatrix} + I_3\begin{bmatrix} 0 \\ 0 \\ 3 \\ \frac{1}{4}(n-9) \end{bmatrix} + I_4\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{4}(n-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ \frac{1}{4}(n-5) \end{bmatrix}$$

satisfies $Ax \geq b$ provided

$$\tfrac14(n-5) \geq \tfrac49 \cdot \tfrac14(n-9) + f_4 \tfrac14(n-1).$$

Since $f_3 + f_4 - \frac49 + f_4 \leq 1$, $f_4 \leq \frac59$. But f_4 is an integer over 9 so $f_4 \leq \frac59$. Hence, $Ax \geq b$ provided

$$\tfrac14(n-5) \geq \tfrac49 \cdot \tfrac14(n-9) + \tfrac59 \cdot \tfrac14(n-1),$$

or $n \geq 13$.

The cases $n = 5$ and $n = 9$ can be directly checked.

(iii) $f_3 = \frac59$.

Then $f_4 \leq \frac49$, and the solution

$$n\begin{bmatrix} 1 \\ 0 \\ 0 \\ \frac12(n-1) \end{bmatrix} - \binom{n}{2}\begin{bmatrix} 0 \\ 1 \\ 1 \\ \frac12(n-5) \end{bmatrix} + f_3\begin{bmatrix} 0 \\ 0 \\ 3 \\ \frac14(n-9) \end{bmatrix} + f_4\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac14(n-1) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2 \\ \frac14(n-9) \end{bmatrix}$$

satisfies $Ax \geq b$ provided

$$\tfrac14(n-9) \geq \tfrac59 \cdot \tfrac14(n-9) + \tfrac49 \cdot \tfrac14(n-1),$$

or $n \geq 25$.

The values $n = 13, 17, 21$ can be checked to see that they do not give $f_3 = \frac59$. The proof for $n \equiv 1 \pmod 4$ is completed.

The case $n \equiv 2 \pmod 4$ has an optimum linear programming basis

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ \frac12(n-2) & \frac12(n-2) & \frac12(n-6) & \frac12(n-2) \end{bmatrix}$$

with primal solution

$$\frac{n}{2},\quad \binom{n}{2},\quad \frac12\binom{n}{3},\quad \tfrac18 n^2(n-7),$$

and dual solution

$$0,\quad 0,\quad \frac{2}{n-2},\quad \frac{4}{n-2}.$$

If $n \geq 10$, the primal solution is non-negative. The case $n = 6$ can be shown to have the edge coloring property. Also, because n is even the primal solution has at most one fractional value so the result follows easily in this case.

The objective is

$$\Delta(\mathrm{cl}(K_n^3)) + \frac{n(n-1)}{6}.$$

4. The complete h-partite hypergraph

K_n^h is a generalization of the complete graph K_n; now we can also generalize the complete bipartite graph $K_{p,q}$. The *complete h-partite hypergraph* $K^h_{n_1,n_2,\ldots,n_h}$ is defined by h disjoint sets $X_1, X_2, \ldots, X_h$, with

$$|X_i| = n_i \quad (1 \le i \le h)$$

$$0 < n_1 \le n_2 \le \cdots \le n_h.$$

The vertex-set is the union $\bigcup X_i$, and $E \subseteq \bigcup X_i$ is an edge if and only if

$$|E \cap X_i| = 1 \quad (1 \le i \le h).$$

Lemma. *The complete h-partite hypergraph has the edge coloring property.*

This was proved by Berge [2].

Theorem 3. *The hereditary closure of the complete h-partite hypergraph has the edge coloring property.*

Proof. Let $H \subseteq \mathrm{cl}(K^h_{n_1,n_2,\ldots,n_h})$ be the hereditary closure of the complete h-partite hypergraph on $X_1, X_2, \ldots, X_h$. Consider h points $a_1, a_2, \ldots, a_h$ which are not in $\bigcup X_i$, and put $X'_i = X_i \cup \{a_i\}$. We shall construct a complete h-partite hypergraph H' on $X'_1, X'_2, \ldots, X'_h$ as follows: For each edge E of H there is an edge E' of H' defined by:

$$E' = E \cup \{a_i \,/\, E \cap X_i = \emptyset\}.$$

Hence, $H' = K^h_{n_1+1,n_2+1,\ldots,n_h+1}$, and there is a bijection between the edges of H and the edges of H'. By the lemma, the complete h-partite hypergraph has the edge coloring property, hence:

$$q(H') = \Delta(H') = (n_2+1)(n_3+1)\cdots(n_h+1).$$

Consider a coloring of the edges of H' into $q(H')$ colors; if we color each edge of H with the same color as the corresponding edge of H', two intersecting edges of H will have different colors; hence

$$q(H) \le q(H').$$

A vertex $x_1 \in X_1$ is of minimum degree in H' and its degree in H' is the same as its degree in H, hence

$$\Delta(H') \le \Delta(H).$$

Thus, we have

$$q(H) \leq q(H') = \Delta(H') \leq \Delta(H) \leq q(H).$$

Hence $\Delta(H) = q(H)$, and therefore H has the edge coloring property.

References

[1] Z. Baranyai, On the factorization of the complete uniform hypergraph, in: A. Hajnal, R. Rado, V.T. Sós, eds. Infinite and Finite Sets, (North-Holland, Amsterdam, 1975) 91–108.

[2] C. Berge, Nombres de coloration de l'hypergraphe h-parti complet, Annali di Mat. Pura et App. (IV), 103 (1975) 3–9. (For general definitions, see: C. Berge, Graphs et Hypergraphs (North Holland, Amsterdam, 1973).)

[3] J.C. Bermond, Private communication.

[4] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem, Operations Res. 9 (1961) 849–859, 11 (1963) 863–888.

[5] E. Lucas, Récréations Mathématiques II (1892). New edition (Blanchard, Paris, 1960) p. 195.

[6] R. Peltesohn, Das Turnierproblem für Spiele zu je dreien, Inaugural Dissertation, Friedrich-Wilhelms Universität, Berlin, 1936.

Annals of Discrete Mathematics 1 (1977) 79-97

SHARP LOWER BOUNDS AND EFFICIENT ALGORITHMS FOR THE SIMPLE PLANT LOCATION PROBLEM

Ole BILDE
Institute of Mathematical Statistics and Operations Research, Technical University of Copenhagen, Copenhagen, Denmark

Jakob KRARUP
Institute of Datalogy, University of Copenhagen, Copenhagen, Denmark

A conceptually straightforward method for generating sharp lower bounds constitutes the basic element in a family of efficient branch and bound algorithms for solving simple (uncapacitated) plant location problems and special versions hereof including set covering and set partitioning.

After an introductory discussion of the problem formulation, a theorem on lower bounds is established and exploited in a heuristic procedure for maximizing lower bounds. For cases where an optimal solution cannot be derived directly from the final tableau upon determination of the first lower bound, a branch and bound algorithm is presented together with a report on computational experience.

The lower bound generation procedure was originally developed by the authors in 1967. In the period 1967-69 experiments were performed with various algorithms for solving both plant location and set covering problems. All results appeared in a series of research reports in Danish and attracted accordingly limited attention outside Scandinavia. However, due to their simplicity and high standard of performance, the algorithms are still competitive with more recent approaches. Furthermore, they have appeared to be quite powerful for solving problems of moderate size by hand.

1. Introduction

Initially, we formulate and discuss the close relationship among three problems, the *simple plant location problem* (PLP), the *set covering problem* (SCP), and the *set partitioning problem* (SPP). Since SCP and SPP can be viewed upon as special cases of PLP, the remaining part of the paper is devoted entirely to PLP. Section 3 deals with a theorem on lower bounds followed by a heuristic procedure for solving the lower bound maximization problem. The bounding procedure is briefly discussed in terms of Lagrangian relaxation in Section 4 before we proceed with a few remarks as to how PLP's may be solved by hand in Section 5. More expedient techniques are, for obvious reasons, required for larger problems. A branch and bound algorithm is presented in Section 6 together with a report on computational results in the concluding Section 7.

The last decade had witnessed a significant research into these problems, not in the least due to their wide applicability to real-world problems. An excellent entrance to the relevant literature is an extensive bibliography, compiled and commented by Francis and Goldstein [11]. Their list comprises 226 papers on normative approaches to location problems published in the period 1963–73.

The more significant works on PLP include, in chronological and alphabetical order, Balinski [4], Bergendahl [6], Efroymson and Ray [10], and Spielberg [18]. Besides, the important paper by Khumawala [16] should be included in the list as being a representative of the current state of art in solving PLP's. Useful references to SCP and SPP are the joint works by Garfinkel and Nemhauser [12, 13] and the series of papers by Balas and Padberg [1, 2, 3]. Readers concerned with real-world problems will find a comprehensive bibliography of applications of SCP and SPP (with emphasis on the latter) in an appendix to [3]; 44 references are cited.

Our personal contributions to PLP, SCP and SPP have manifested themselves in [6, 7, 17] plus various lecture notes, unfortunately for most readers, with several of the more important sections in Danish. The present paper, now in a language accessible to wider circles, is basically an extract of those earlier works except for the references above to more recent conquests and the inclusion of Section 4 on Lagrangian relaxation. It is our sincere belief that the computational efficiency and the simplicity of our approach (which makes it suited for hand computation as well) will justify this apparent reboiling of old bones.

In addition, some of the "open" questions raised in our earlier contributions have given rise to a 1975-paper [8] where PLP's with certain structures are studied. The main results comprise a polynomially bounded algorithm and the establishment of a connection to linear programming such that post-optimal analysis of LP is directly applicable for a class of structured PLP's. Since the basic principles underlying this new step ahead still are those from 1967–69, it is conceivable that other researchers also may find some inspiration for future work. Anyway, the PLP-SCP-SPP-family still offers lots of challenges!.

2. Plant location, set covering and set partitioning

The so-called *simple plant location problem* deals with the supply of a single commodity from a subset of plants (sources) to a set of customers (sinks) with a prescribed demand for the commodity. Irrespective of its realism in practice, we assume unlimited capacity of each plant, i.e. any plant can satisfy all demands.[1] Given the cost structure, we seek a minimum cost transportation plan which satisfies the demand at each customer.

The constituents of a PLP are:

m : the number of potential plants indexed by i, $i \in I = \{1, 2, \ldots, m\}$;

n : the number of customers indexed by j, $j \in J = \{1, \ldots, n\}$;

k_i : fixed cost associated with plant i;

[1] The adjective *simple* has been coined by Spielberg [18] to express the assumption of unlimited capacities. In this context, simple has now become commonly accepted as synonymous with *uncapacitated*.

b_j: demand (number of units) at customer j;
t_{ij}: unit transportation cost from plant i to customer j.

We shall frequently use the adjectives "open" and "closed" for designating the state of a plant. The cost of sending no units from a plant is zero (i.e., the plant is "closed") while any positive shipment from the i'th plant incurs a fixed cost k_i (the plant is "open") independent of the quantity shipped, plus a cost t_{ij} proportional to the number of units transported to the j'th customer.

We may adopt the mixed-integer mode formulation due to Balinski [4] but an immediate observation (also mentioned by Efroymson and Ray [10]) leads directly to an all-integer formulation in 0–1 variables:

Let $y_i = 1$ if plant i is open; otherwise $y_i = 0$. For any set of y's, the optimal transportation plan can be determined directly by assigning each customer to the "nearest" open plant, provided that at least one plant is open.

This implies that we may restrict ourselves to considering solutions where every customer is supplied only by a single plant. Accordingly, let $x_{ij} = 1$ if customer j is supplied by plant i; otherwise, $x_{ij} = 0$. Furthermore, let $c_{ij} = t_{ij}b_j$ denote the total transportation cost incurred by $x_{ij} = 1$. Individual production costs (if any) at each plant can easily be incorporated; if p_i is the unit production cost associated with plant i, we may replace $t_{ij}b_j$ by $(p_i + t_{ij})b_j$ in the expression for calculating the c_{ij}'s.

We observe finally, that possible negative fixed costs do not present anything new. Without loss of generality, we shall therefore assume all fixed costs to be nonnegative. We shall also assume nonnegative c_{ij}'s.

The simple *plant location problem* can now be stated (PLP):

$$\sum_{i=1}^{m}\Big(k_i y_i + \sum_{j=1}^{n} c_{ij}x_{ij}\Big) = z_{PLP}(\min)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \text{all } j \tag{1}$$

$$\left.\begin{array}{ll} y_i - x_{ij} \geq 0, & \\ x_{ij} = 0, 1; & y_i = 0, 1 \end{array}\right\} \text{ all } i, j$$

Like the Classical Transportation Problem, the underlying network for PLP is $K_{m,n}$, the complete bipartite network with m sources, n sinks and $m \times n$ edges. Of more significance are the deviations between these two problems: the unlimited supply at each source and, in particular, the nonnegative fixed costs. The presence of the latter yields a concave cost function (with a discontinuity at zero for every source); hence, local optima different from the global may occur. Therefore, we cannot advocate the use of techniques based on extensions of Linear Programming (e.g. separable programming): on the contrary, experiments have shown that the results obtained may be quite misleading. Examples (or warnings against local optima) can be found in Bergendahl [5]. Rather, studies of PLP's can be claimed to be a topic of combinatorial programming.

Before proceeding with PLP, let us introduce two more problems belonging to the same family.

Let $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ denote two finite sets and let $A = \{a_{ij}\}$ be a $m \times n$ matrix of zeros and ones.

A subset $\bar{I} \subseteq I$ defines a *cover* of J if

$$\sum_{i \in \bar{I}} a_{ij} \geq 1, \quad \text{all } j. \tag{2}$$

$\bar{I}$ is called a *partition* of J if

$$\sum_{i \in \bar{I}} a_{ij} = 1, \quad \text{all } j. \tag{3}$$

Let $y_i = 1$ if $i \in \bar{I}$ and 0 otherwise and let k_i be the cost associated with the i'th row of A. The *set covering problem* is to find a cover of minimum cost (SCP):

$$\sum_{i=1}^{m} k_i y_i = z_{SCP}(\min)$$

$$\sum_{i=1}^{m} a_{ij} y_i \geq 1, \quad \text{all } j \tag{4}$$

$$y_i = 0, 1, \quad \text{all } i.$$

Accordingly, the *set partitioning problem* reads (SPP):

$$\sum_{i=1}^{m} k_i y_i = z_{SPP}(\min)$$

$$\sum_{i=1}^{m} a_{ij} y_i = 1 \quad \text{all } j \tag{5}$$

$$y_i = 0, 1 \quad \text{all } i.$$

Any SPP having a feasible solution can be converted into a SCP by changing the cost vector. Independent verifications of this postulate can be found in Bilde [6] and the perhaps more accessible book by Garfinkel and Nemhauser [13, p. 300].

Now consider a particular PLP with all c_{ij}'s equal to zero or infinity and define a SCP with the same k_i's and with

$$a_{ij} = \begin{cases} 1, & \text{if } c_{ij} = 0, \\ 0, & \text{if } c_{ij} = \infty, \end{cases} \quad \text{all } i, j. \tag{6}$$

Conversely, for any SCP, define a PLP with the same k_i's and

$$c_{ij} = \begin{cases} 0, & \text{if } a_{ij} = 1, \\ \infty, & \text{if } a_{ij} = 0, \end{cases} \quad \text{all } i, j. \tag{7}$$

The existence of a finite solution to PLP implies the existence of a feasible solution to SCP and vice versa. For any such pair of solutions, both objective functions will assume the same value.

Thus, SCP (and SPP) can be considered as special cases of the more general PLP. The following sections — dealing entirely with PLP — will therefore apply for SCP and SPP as well.

A word about the computational complexity of PLP's: Karp's Main Theorem [15] states that 21 computational problems — virtually comprising all combinatorial optimization problems — are *NP-complete*. Verbally, it means that either each of them is solvable by a polynomial-bounded algorithm (i.e. an algorithm which terminates within a number of steps bounded by a polynomial in the length of the input) or none of them is. SCP is on that list too, and due to the relationship between PLP and SCP, we can conclude that PLP is NP-complete as well.

3. Lower bounds for PLP

Consider the PLP formulation (1) where the objective is to *minimize* total cost. A direct way of generating a *lower* bound on $z^0_{PLP} = \min\{z_{PLP}\}$ could be to *relax* the integrality constraints by replacing

$$x_{ij}, y_i = 0, 1 \quad \text{by} \quad x_{ij}, y_i \geq 0, \quad \text{all } i, j$$

and to solve the resulting LP-problem.

However, due to reasons to be discussed later, we shall desist from use of an LP-technique; instead a less sophisticated but highly effective heuristic method for the lower bound maximization problem is suggested. The exposition follows the lines given in Bilde and Krarup [7].

Let $\Delta = \{\Delta_{ij}\}$ be a $(m \times n)$-matrix of reals. Δ is said to be *feasible* if the following two conditions are met

$$\begin{aligned} &\sum_{j=1}^{n} \Delta_{ij} \leq k_i, \quad \text{all } i, \\ &\Delta_{ij} \geq 0, \quad \text{all } i, j. \end{aligned} \tag{8}$$

By introducing Δ in the PLP-formulation (1) by adding and subtracting the same expression in the objective function, we arrive at an equivalent formulation:

$$\begin{aligned} &\sum_{i=1}^{m} \Big(k_i y_i - \sum_{j=1}^{n} \Delta_{ij} x_{ij}\Big) + \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij} + \Delta_{ij}) x_{ij} = z_{PLP}(\min) \\ &\sum_{i} x_{ij} = 1 \quad \text{all } j \\ &\left.\begin{array}{l} y_i - x_{ij} \geq 0 \\ x_{ij} = 0, 1; \ y_i = 0, 1 \end{array}\right\} \quad \text{all } i, j. \end{aligned} \tag{9}$$

The optimal solution to (1) or (9) is denoted by (x^0, y^0) with $\min\{z_{PLP}\} = z^0_{PLP}$.

For the individual terms in the left hand part of the transformed objective function, we have

$$k_i y_i - \sum_{j=1}^{n} \Delta_{ij} x_{ij} \geq 0, \quad \text{all } i \tag{9a}$$

for any feasible Δ_{ij} and for any (x, y) representing a feasible solution to (9).

For any *fixed* set of feasible Δ_{ij}'s, designate the LP-problem (LBPLP)

$$\begin{aligned} &\sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij} + \Delta_{ij}) x_{ij} = z_{LBPLP} (\min) \\ &\sum_{i=1}^{m} x_{ij} = 1, \quad \text{all } j \\ &x_{ij} \geq 0, \quad \text{all } i, j \end{aligned} \tag{10}$$

with $\min\{z_{LBPLP}\} = z^*_{LBPLP}$.

For $(x, y) = (x^0, y^0)$, we obtain by means of (9a)

$$\begin{aligned} z^0_{PLP} &= \sum_{i=1}^{m} \left(y_i^0 k_i - \sum_{j=1}^{n} \Delta_{ij} x^0_{ij} \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij} + \Delta_{ij}) x^0_{ij} \\ &\geq \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij} + \Delta_{ij}) x^0_{ij} \geq z^*_{LBPLP}. \end{aligned} \tag{11}$$

No sophistication is required for solving LBPLP. By inspection of (10), we realize that

$$z^*_{LBPLP} = \sum_{j=1}^{n} \min_i \{c_{ij} + \Delta_{ij}\} \tag{12}$$

i.e. z^*_{LBPLP} is simply the sum of the column minima of the $(C + \Delta)$ – matrix. This explains the prefixed letters LB (Lower Bound) in LBPLP and proves the following

Theorem 1.

$$\sum_{j=1}^{n} \min_i \{c_{ij} + \Delta_{ij}\} \leq z^0_{PLP}$$

where Δ_{ij} is any set of nonnegative numbers satisfying

$$\sum_{j=1}^{n} \Delta_{ij} \leq k_i, \quad \text{all } i.$$

Verbally, Theorem 1 asserts that a lower bound on z^0_{PLP} can be achieved as the summed column minima of the $(C + \Delta)$-matrix for any set of feasible Δ_{ij}'s. Such a lower bound is, of course, strongly dependent on the way in which Δ is determined.

According to Theorem 1, the *sharpest* lower bound w^0_{LBPLP} is found as the optimal solution to

$$\sum_{j=1}^{n} \min_i (c_{ij} + \Delta_{ij}) = W_{\ldots}(\max),$$

$$\sum_{j=1}^{n} \Delta_{ij} \leq k_i, \quad \text{all } i, \tag{13}$$

$$\Delta_{ij} \geq 0, \quad \text{all } i, j.$$

Actually (13) could be slightly reformulated and solved by means of some LP technique. But since lower bounds normally have to be generated repeatedly throughout the computations in a branch and bound algorithm, we seek a bounding procedure which—rather than striving after an optimal solution to the bounding problem—combines sharp bounds with limited computational effort.

The following heuristic procedure which possesses both properties is initiated with the given C-matrix and a Δ-matrix consisting entirely of zeros. By introducing a set r_i of auxiliary variables, defined by the differences

$$r_i = k_i - \sum_{j=1}^{n} \Delta_{ij}, \quad \text{all } i, \tag{14}$$

r_i must equal k_i initially and the $n+1$ numbers $(r_i, \Delta_{i1}, \ldots, \Delta_{in})$ can throughout the computations be viewed upon as a *partitioning* of the corresponding k_i.

The idea of the procedure is to find partitionings of the fixed costs so as to maximize the summed column minima of the resulting $(C+\Delta)$-matrix. While all c_{ij}'s preserve their original values, the elements of Δ are increased iteratively such that any augmentation of some Δ_{ij} is followed by a reduction of the corresponding r_i by the same amount.

The procedure operates on the columns in the $(C+\Delta)$-matrix, one at a time. In each step we select a column and attempt to alter a subset of its elements by increasing the respective Δ_{ij}'s in a way which, so to speak, gives maximum effect on the corresponding column minimum with a minimum "consumption" of the r_i's involved.

A few observations: To increase an element which is not a column-minimum in the actual $(C+\Delta)$-matrix will not influence that column minimum. Furthermore, if two or more elements in a column are equal to the column-minimum, no effect on the lower bound will be obtained unless they are all increased. Finally, we shall see that a column-minimum cannot be further increased if any of the auxiliary variables involved have been reduced to zero.

In order to guide the search for the column to be the next candidate for further augmentation, we associate a so-called *level-number*, λ_j, with the j'th column in $C+\Delta$ which is equal to the number of occurrences of the smallest element in that column. At any stage of computation, the next candidate for selection j^* is the column with the smallest level number. In case of a tie, that column with the smallest index is chosen. So, the selection rule reads:

$$j^* = \min\left\{ j \mid \lambda_j = \min_{k \in J} \{\lambda_k\} \right\}. \tag{15}$$

Instead of proceeding with the formal exposition of the heuristic approach which will require additional symbols, we shall illustrate the method by means of a numerical example.

A PLP ($m = 5$, $n = 4$) is given in Table 1; also the initial values of the level numbers are shown.

Table 1

Plant no.	Fixed costs k_i	Customer 1	2	3	4	
1	6	2	10	9	8	$C' = C + \Delta$
2	6	8	7	9	2	
3	7	9	7	6	2	
4	3	7	10	7	5	
5	5	1	9	15	4	
	Level numbers, λ_j	1	2	1	2	

Initial tableau: $\Delta_{ij} = 0$, all i, j; $r_i = k_i$, all i.

Step 1: $\min_{j \in J}\{\lambda_j\} = 1 = \lambda_1 = \lambda_3$; $j^* = \min\{1, 3\} = 1$.
$c_{51} = 1$ is the smallest element in the first column and $c_{11} = 2$ is the second smallest. We intend to choose Δ_{51} so as to increase $c_{51} + \Delta_{51}$ as much as possible without exceeding c_{11}, i.e. by an amount $\beta = c_{11} - c_{51} = 2 - 1 = 1$. r_5 shall remain nonnegative upon reduction by the same amount which, in this case does not affect the value of β, i.e.

$$\beta = \min\{(c_{11} - c_{51}), r_5\} = \min\{1, 5\} = 1$$

Finally, to complete the updating of the tableau after the first step increase λ_1 by 1 resulting in Table 2.

Table 2

	i	r_i	1	2	3	4	
	1	6	2	10	9	8	$C + \Delta$
	2	6	8	7	9	2	
$\beta = 1$	3	7	9	7	6	2	
	4	3	7	10	7	5	
	5	4	2	9	15	4	
		λ_j	2	2	1	2	

In general, all elements which have been changed from one step to the next are shown underlined in the corresponding tableau.

Step 2: $j^* = 3$. Increase Δ_{33} and reduce r_3 by $\beta = \min\{(c_{34} + \Delta_{34}) - (c_{33} + \Delta_{33}), r_3\} = \min\{1, 7\} = 1$, and update λ_3, producing Table 3.

Table 3

	i	r_i	1	2	3	4	
	1	6	2	10	9	8	
	2	6	8	7	9	2	
$\beta = 1$	3	6	9	7	7	2	$C + 1$
	4	3	7	10	7	5	
	5	4	2	9	15	4	
		λ_j	2	2	2	2	

Step 3: $j^* = 1$. Increase Δ_{11} *and* Δ_{51} and reduce r_1 *and* r_5 by $\beta = \min\{(c_{j1} + \Delta_{j1}) - (c_{11} + \Delta_{11}), r_1, r_5\} = \min\{5, 6, 4\} = 4$.

Note that the *effective* upper bound on β is r_5 ($= 4$ before reduction). Accordingly, λ_1 remains unaltered and, what is more important, the smallest element in the first column can not be further increased since $r_5 = 0$ after reduction. To emphasize this, we mark row 5 and column 1 in Table 4 with asterisks.

Table 4

		i	r_i	1*	2	3	4	
		1	2	6	10	9	8	
		2	6	8	7	9	2	
$\beta = 4$		3	6	9	7	7	2	$C + 4$
		4	3	7	10	7	5	
	*	5	0	6	9	15	4	
			λ_j	2	2	2	2	

Step 4: A marked column is no longer a potential candidate for selection; hence $j^* = 2$.

Table 5.
$\beta = \min\{(9-7), 6, 6\} = 2$

		i	λ_i	1	2	3	4	
		1	2	6	10	9	8	
		2	4	8	9	9	2	
$\beta = 2$		3	2	9	9	7	2	$C = 4$
		4	3	7	10	7	5	
	*	5	0	6	9	15	4	
			λ_j	2	3	2	2	

One of the smallest elements of column 2 in Table 5 appears in a marked row and can accordingly never be further increased. Column 2 is therefore marked as well.

Step 5: $j^* = 3$; $\beta = \min\{(9-7), 4, 3\} = 2$ produces Table 6.

Table 6

		i	λ_i	1	2	3	4	
		1	2	6	10	9	8	
		2	4	8	9	9	2	
$\beta = 2$		3	2	9	9	9	2	$C = 4$
		4	1	7	10	9	5	
	*	5	0	6	9	15	4	
			λ_j	2	2	4	2	

Step 6: $j^* = 4$; $\beta = \min\{(4-2), 4, 2\} = 2$

Table 7

		i	λ_i	1	2	3	4	
		1	2	6	10	9	8	
		2	2	8	9	9	2	
$\beta = 2$	*	3	0	9	9	9	2	$C = 4$
		4	1	7	10	9	5	
	*	5	0	6	9	15	4	
			λ_j	2	2	4	2	

c_4 is reduced to zero; consequently, row 3 and column 4 are both marked in Table 7. Also the last column (column 3) can be marked since one of its smallest elements now appears in a marked row.

All columns are marked and the process terminates with a lower bound equal to the sum of the smallest elements in each column.

$$\sum_{j=1}^{4} \min_i \{c_{ij} + \Delta_{ij}\} = 6 + 9 - 9 + 4 \le z^*_{\mathrm{PLP}}. \tag{16}$$

4. The bounding procedure and Lagrangian relaxation

For a general integer LP-problem (IP):

$$\begin{aligned} &cx = z_{\mathrm{IP}}(\min), \\ &Ax \ge b,\ Bx \ge d, \\ &x \ge 0;\ x_i \text{ integer},\ i \in I, \end{aligned} \tag{17}$$

the *Lagrangian relaxation* of IP relative to the constraint set $Ax \ge b$ and a conformable nonnegative vector λ is defined by IPR:

$$\begin{aligned} &cx + \lambda(b - Ax) = z_{\mathrm{IPR}}(\min), \\ &Bx \ge d, \\ &x \ge 0;\ x_i \text{ integer},\ i \in I. \end{aligned} \tag{18}$$

The idea of Lagrangian relaxation is to identify a set of "complicating constraints" (here $Ax \ge b$), weighting these by multipliers and inserting them in the objective function in order to obtain a problem IPR which, hopefully, is simpler to solve than the underlying problem IP.

A general theory of Lagrangian relaxation, which has provided a unifying framework for several bounding procedures in discrete optimization, has been developed by Geoffrion [14] with particular emphasis on applications in the context of LP-based branch and bound.

Let z^*_{IPR} denote the minimum value of z_{IPR} for given λ. In a discussion of the potential usefulness of a Lagrangian relaxation, Geoffrion points out that the ideal choice would be to take λ as an optimal solution to

$$\max_{\lambda \ge 0} \{z^*_{\mathrm{IPR}}\}, \tag{19}$$

which is the formal Lagrangian dual of IP with respect to the constraints $Ax \ge b$.

Now let us exemplify the situation sketched above by reconsidering our PLP with $y - x_{ij} \ge 0$ as the set of "complicating constraints" and with the Δ_{ij}'s as the corresponding set of nonnegative multipliers. By (1), (17) and (18), the derived Lagrangian problem becomes (PLPR):

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+\Delta_{ij})x_{ij}-\sum_{i=1}^{m}\Big(k_i-\sum_{j=1}^{n}\Delta_{ij}\Big)y_i=z_{PLPR}(\min),$$

$$\sum_{i=1}^{m}x_{ij}=1,\quad \text{all } j, \tag{20}$$

$$x_{ij}=0,1,\;\; y_i=0,1,\quad \text{all } i,j.$$

If we restrict ourselves to considering multipliers satisfying (8), the y_i-variables may be removed from the Lagrangian problem because $\Sigma_j\Delta_{ij}=k_i$ for all i, and PLPR above coincides with (10) which is solvable by inspection.

Due to (12), the minimum value of z_{PLPR} is determined by

$$z^0_{PLPR}=\sum_{j=1}^{n}\min_i\{c_{ij}+\Delta_{ij}\}.$$

In terms of Lagrangian relaxation, our approach can be viewed upon as a parametrized relaxation where the bounding procedure is a rule for setting the Δ_{ij}-parameters to obtain sharp lower bounds.

In this context it is of interest to notice that the Lagrangian dual (19) and the lower bound maximization problem (13) are equivalent.

As was mentioned in the concluding remarks of Section 2 on computational complexity, PLP is NP-complete. Since it is very unlikely that a polynomial-bounded algorithm can be devised for a NP-complete problem (e.g. a PLP), it is reasonable to advocate the use of heuristics for solving large-scale PLP's. This is one of the main arguments for Cornuejols, Fisher and Nemhauser [9] for studying heuristics for solving a so-called *account location problem* which, as they point out, is mathematically equivalent to PLP. Their main results are on the *quality of solutions* obtained from heuristics and the *quality of bounds* obtained from LP and Lagrangian relaxation. It is interesting to realize the fact that the question on *how* the subset of "complicating constraints" should be selected does not necessarily have an obvious answer. While our Lagrangian relaxation is relative to $y_i - x_{ij} \geq 0$, the complementary subset $\Sigma_i x_{ij}=1$ is applied in [9]. However, to make a detailed comparison of our ideas to those in [9] and to compare the computational experience must be left over as an appropriate subject for future research.

5. Solving PLP's by hand

Let (r^*, Δ^*) denote the final values of (r, Δ) upon termination of the bounding procedure as was described in Section 3 and let w^* be the lower bound thus obtained:

$$r_i^*=k_i-\sum_{j=1}^{n}\Delta_{ij}^*,\quad \text{all } i \tag{21}$$

$$
\begin{aligned}
w^* &= \sum_{j=1}^{n} \alpha_j \le w^0_{\mathrm{MILP}} \le z^0_{\mathrm{MIP}} \\
&\text{where} \quad \alpha_j = \min_i \{c_{ij} + \Delta^*_{ij}\}, \quad \text{all } j.
\end{aligned}
\tag{22}
$$

It may occur that equality holds, not only between w^* and w^0_{MILP} but also between the latter and z^0_{MIP}. To illustrate this, let

$$D_j = \{i \mid \gamma^*_i = 0 \wedge (c_{ij} + \Delta^*_{ij}) = \alpha_j\}, \quad \text{all } j \tag{23}$$

and suppose a subset $P \subseteq I$ of plants can be found which satisfies the following conditions:

$$P \subseteq \bigcup_{j=1}^{n} D_j \tag{24}$$

$$P \cap D_j \neq \emptyset, \quad \text{all } j \tag{25}$$

$$\sum_{j=1}^{n} \Delta^*_{ij} \Delta^*_{sj} = 0, \quad \text{all } (i, s),\ i \in P,\ s \in P,\ i \neq s. \tag{26}$$

Define $y_i = 1$ if $i \in P$; otherwise, $y_i = 0$. For each j, select an entry

$$\{(i, j) \mid i \in P \cap D_j,\ \Delta^*_{ij} > 0\} \tag{27}$$

or if no such entry exists, let (i, j) be any member of the nonempty subset

$$\{(i, j) \mid i \in P \cap D_j,\ \Delta^*_{ij} = 0\} \tag{28}$$

and assign the value 1 to the corresponding x_{ij} while all remaining x_{sj}'s, $s \neq i$ are kept at zero level.

Intuitively, (23)–(25) means that we seek a subset P of open plants for which the fixed costs have been totally absorbed during the process of constructing the Δ^*_{ij}'s. Furthermore, all $x_{ij} = 1$ selected by (27) or (28) corresponds to entries (i, j) with $y_i = 1$ and $c_{ij} + \Delta^*_{ij} = \alpha_j$, and (26) secures that $\Delta^*_{ij} > 0$, $\Delta^*_{sj} > 0$ cannot occur simultaneously for any pair (i, s) of open plants, i.e.

$$\sum_{j \in J^i} \Delta^*_{ij} = k_i,\ J^i = \{j \mid x_{ij} = 1\},\ \forall i \in P.$$

Besides, the column minima of every column of $C + \Delta^*$ must appear in at least one of the rows comprised by P. Thus, we have constructed not only a feasible solution (x, y) representing an upper bound on z^0_{MIP} but a solution for which the lower bound w^* coincides with the value of the objective function. Hence, (x, y) is optimal.

Consider the final tableau obtained in Section 3, now with all elements of $C + \Delta^*$ written explicitly as the sum of two terms displayed in Table 8.

Table 8

i	k_i	r_i	1	2	3	4	
1	6	7	7 + 1	10 + 0	9 + 0	8 + 0	
2	6	2	8 + 0	7 + 2	5 + 0	2 + 2	
3	7	0	8 + 0	$\boxed{7 + 2}$	$\boxed{6 + 3}$	$\boxed{2 + 2}$	$C = 47$
4	3	1	6 + 0	10 + 8	7 + 2	5 + 0	
5	5	0	$\boxed{1 + 5}$	$\boxed{9 + 0}$	6 + 0	$\boxed{4 + 0}$	
Column minima α_j			6	9	5	4	

By means of (23): $D_1 = \{5\}$, $D_2 = \{3, 5\}$, $D_3 = \{3\}$, $D_4 = \{3, 5\}$. Obviously, $P = \{3, 5\}$ satisfies (24)–(26); hence $y_3 = y_5 = 1$. The complete solution with $w^* = \sum \bar{c}_{ij} = 28$ is achieved by (27) and (28); actually, (27) suffices for all columns in this particular case: $x_{51} = x_{52} = x_{33} = x_{54} = 1$.

Note, that we have *not* claimed the existence in general of a subset P satisfying (24)–(26). Although it occurs frequently in practice, some reflection will show that counter examples are easily constructed for any $m \geq 3$, $n \geq 3$.

However, searching for P by simple inspection of the final bounding tableau is not an overwhelming task for problems of moderate size. A series of real-world problems ranging up to $m = 29$, $n = 14$ were optimally solved on a blackboard (including determination of the lower bound) within a few minutes.

In cases where P cannot be derived directly from the tableau, a branch and bound technique almost suggests itself as the most natural way to proceed. For problems of a reasonable size, hand computation is still a possibility provided that the lower bounds are generated as described in Section 3.

6. Branch and bound algorithms

Several experiments with different selection rules for branching were performed by the authors in the period 1967–69. Some preliminary results are reported on in Krarup [17] but the methods were further improved later on by Bilde [6] from which the material in this section is extracted.

At any node t in the branch and bound tree (representing a subproblem of the given PLP or the subset of feasible solutions to that subproblem) the set of plants is partitioned into three subsets

"Open" plants: $I_1^t = \{i \mid y_i = 1$ at node $t\}$,

"Closed" plants: $I_0^t = \{i \mid y_i = 0$ at node $t\}$,

"Free" plants: $I_2^t = \{i \mid y_i$ is undefined at node $t\}$.

Any fixed y_i at node t (i.e. $i \in I_0^t \cup I_1^t$) preserves its value when further branchings from node t are performed.

For node t, define a subproblem of the original PLP by *ignoring*:

all plants $i \in I_0^t$ (computationally, we may delete the corresponding rows or replace k_i, $i \in I_0^t$, by very large numbers)

all fixed costs associated with the subset of open plants (k_i, $i \in I_1^t$ are replaced by zeros).

Let w_0^t denote the lower bound obtained by the bounding procedure for this particular subproblem. Obviously, a lower bound w^t for the subset of solutions represented by node t can be achieved as

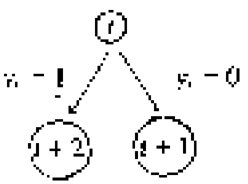
$$w^t = w_0^t + \sum_{i \in I_1^t} k_i. \tag{29}$$

No further branching from node t is required in the following two cases:

(1) If $I_2^t = \emptyset$, an optimal solution determined by (I_0^t, I_1^t) has been found for node t with $z_{\mathrm{PLP}} = w^t$.

(2) If $w^t \geq z_{\mathrm{BS}}$, where z_{BS} represents the value of the best solution so far and where this solution is obtained from the bounding procedure by opening those plants (rows) which are marked and by serving all customers from the "nearest" opened plants.

On the other hand, if $I_2^t \neq \emptyset \wedge w^t < z_{\mathrm{BS}}$, two new subproblems corresponding to nodes $(t+1)$ and $(t+2)$ are generated by the branching rule involving the selection of a "free" plant $i \in I_2^t$:



Accordingly,

$$I_1^{t+1} = I_1^t;\quad I_0^{t+1} = I_0^t \cup \{i\};\quad I_2^{t+1} = I_2^t - \{i\}$$

$$I_1^{t+2} = I_1^t \cup \{i\};\quad I_0^{t+2} = I_0^t;\quad I_2^{t+2} = I_2^t - \{i\}.$$

What remains to discuss is the *branching rule* itself: Determination of that node t from which to branch and selection of that free plant $i \in I_2^t$ on which the branching is to be based.

Two different rules are tested in [6]. Both apply n_i and r^3 as was introduced by (14) and (21) respectively.

Whenever a lower bound w^t has been determined by (79), the final bounding tableau contains actual values of r_i^*. For a forthcoming node $t+2$ with some $y_i = 1$, $i \in I_1^t$, the following relation must hold

$$w^{t+2} \geq r_i^* + w^t.$$

Rules A and B can now be stated:

Rule A: Select from I_1^t a plant $\hat{i}$ for which $r_{\hat{i}} = \max_{i \in I_1^t}\{r_i\}$. Perform the branching and proceed with node $(t+1)$.

Rule B: Let $\hat{i}$ denote that row which was the *first* to be marked in the bounding procedure for calculating w^t. Due to the definition of the subproblem represented by node t, $\hat{i}$ is certainly a member of the nonempty subset I_1^t. Select that $\hat{i}$ for the next branching and proceed with node $(t+2)$.

Clearly, we attempt as soon as possible to exclude the "bad" plants by Rule A or to include the "good" plants by Rule B.

7. Computational experience

The two versions of the algorithm were tested in 1969 on an IBM 7094. The results presented below in Tables A and B appeared originally in [6].
Headings of the tables:

m: number of potential plants (rows),
n: number of customers (columns),
w: the first generated lower bound,
S: the number of distinct solutions obtained,
$z^1_{\min}$: the value of the first solution obtained,
$z^0_{\min}$: the value of the optimal (or best) solution,
BF: number of branchings required for obtaining the first solution,
BO: number of branchings required for obtaining an optimal solution,
BT: total number of branchings,
Time: computing time (IBM 7094) including input output (sec.)

Problem a1 is a set covering problem with a density (percentage of ones) equal to 15%.
For all the remaining problems, the elements in the C-matrix have been drawn at random from a discrete uniform distribution over the interval (0, 1000).

Except for problem e10 where all fixed costs are equal to 10,000, the fixed costs for problems in Table 9 are all chosen at random over the intervals (1,000, 10,000) for b1–b3 and (1,000, 2,000) for c1 and c2.

The relative difference between $z^0_{\min}$ and w is 0 (a1, b2) and a few percent (b1, b3) so that the first application of the bounding procedure almost suffices for solving

Table 9
Computational experience with Rule A.

Problem	m	n	w	S	z^*	$\hat{z}$	BF	BO	BT	Time (sec.)
a1	29	24	31	7	31	31	20	20	20	< 1
b1	50	100	15230	1	15372	15372	43	43	46	3
b2	50	100	14020	1	14020	14020	44	44	44	3
b3	50	100	15110	2	15919	15530	44	47	50	8
c1*	50	100	13954	7	17237	15180	44	571	(709)	(> 250)
c2*	50	100	14973	1	16691	16694	43	43	(650)	(> 250)
e10	50	100	41841	3	47868	47012	50	78	(274)	(> 150)

* Execution of the program interrupted before optimality was proved.

Table 10.
Computational experience with Rule B

Problem	m	n	w	S	z^*	$\hat{z}$	BF	BO	BT	Time (sec.)
d1	30	80	12748	3	13815	13416	8	212	216	11
d2	30	80	16989	3	19917	19779	6	24	219	24
d3	30	80	20950	1	23821	23821	5	5	169	19
d4	30	80	24040	2	27559	27512	4	14	141	17
d5	30	80	26502	2	30559	30512	4	8	106	14
d6	30	80	29399	2	33559	33512	4	27	101	15
d7	30	80	31887	1	36122	36122	3	3	85	13
d8	30	80	33970	1	38122	38122	3	3	55	11
d9	30	80	36075	1	40122	40122	3	3	47	11
d10	30	80	37922	1	42122	42122	3	3	43	11
e1	50	100	13054	7	14984	14984	10	19	1271	212
e2	50	100	19519	5	21796	21593	7	50	1112	172
e3	50	100	22546	4	26750	25111	6	56	384	92
e4	50	100	26057	2	30350	30111	5	39	258	65
e5	50	100	29249	2	33712	33573	4	14	195	55
e6	50	100	32024	2	36712	36573	4	9	136	43
e7	50	100	34670	2	39712	39573	4	15	151	42
e8	50	100	37141	2	42712	42573	4	7	143	48
e9	50	100	39725	1	45012	45012	3	3	117	44
e10	50	100	41841	1	47012	47012	3	3	79	37

these problems. However, a considerable growth of the relative difference is noticed for (c1, c2, e10) and Rule A is no longer able to provide optimality within a reasonable amount of time.

With the idea of Rule A in mind, a plausible explanation is that problems (a1, b1–b3) are characterized by very few good solutions while the converse is true for (c1, c2, e10). The conclusion is that the performance of Rule A is good in some situations but unsatisfactory for problems with a "flat" optimum.

Having realized the drawbacks of Rule A, a natural alternative would be to "reverse" the philosophy underlying the selection of that plant on which the next branching is to be based. Accordingly, Rule B was implemented and tested on a series of examples, all believed to represent the most difficult cases: All fixed costs of the same magnitude.

For the twenty problems, d_i and e_i in Table 10, all fixed costs are equal to $1000 \times q$. For e10 which appears in both tables, a substantial drop in computing time is recorded. In general, Rule B seems to be efficient for solving problems with a large number of near-optimal solutions. Note, that the first solution obtained is optimal in 35% of all examples and that the number of branchings required (the BF-column) is extremely low.

Acknowledgements

Sincere thanks are due to P.M. Pruzan and to the referees who provided us with useful suggestions which led to several improvements of the exposition. Furthermore, we are indebted to P.L. Hammer and to K. Spielberg who independently expressed interest in our work at an early stage and encouraged us to perform this extract in English.

References

[1] E. Balas and M.W. Padberg, On the set covering problem, *Operations Res.* 20 (6) (1972) 1152–1161.
[2] E. Balas and M.W. Padberg, On the set covering problem II. An algorithm for set partitioning, *Operations Res.* 23 (1) (1975) 74–90.
[3] E. Balas and M.W. Padberg, Set partitioning, in: B. Roy ed., *Combinatorial Programming: Methods and Applications* (Reidel, Dordrecht, 1975) 205–258.
[4] M.L. Balinski, On finding integer solutions to linear programs, *Mathematica* (1964) 225–248.
[5] G. Bergendahl, *Models for capacity planning*, Ph.D. Dissertation in Business Administration, University of Lund, Sweden (1966).
[6] O. Bilde, *Nonlinear and discrete programming*, Ph.D. Dissertation No. 9, IMSOR, The Technical University of Denmark (1970).
[7] O. Bilde and J. Krarup, *Bestemmelse af optimal beliggenhed af produktionssteder*, Research Report, IMSOR, The Technical University of Denmark (1967).
[8] O. Bilde and J. Krarup, *Plant location, set covering and economic lot size: an O(mn)-algorithm for structured problems*, Research Report, IMSOR, The Technical University of Denmark (1969) and Report No. 75/6, Institute of Datalogy, University of Copenhagen (1975).

[9] G. Cornuejols, M.L. Fisher and G.L. Nemhauser, On the uncapacitated location problem, Research Report No. 7602, CORE, Université Catholique de Louvain (1976); *Ann. Discrete Math.* 1 (1977) 163–177.

[10] M.A. Efroymson and T.L. Ray, A branch-bound algorithm for plant location, *Operations Res.* 14 (3) (1966) 361–368.

[11] R.L. Francis and J.M. Goldstein, Location theory: a selective bibliography, *Operations Res.* 22 (2) (1974) 400–410.

[12] R.S. Garfinkel and G.L. Nemhauser, The set partitioning problem: set covering with equality constraints, *Operations Res.* 17 (5) (1969) 848–856.

[13] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming*, (Wiley, New York, 1972).

[14] A.M. Geoffrion, Lagrangian relaxation for integer programming, *Math. Programming Study* 2 (1974) 82–114.

[15] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–103.

[16] B.M. Khumawala, An efficient branch and bound algorithm for the warehouse location problem, *Management Sci.* 18 (12) (1972) B-718–731.

[17] J. Krarup, *Fixed-cost network flow problems*, Ph.D. Dissertation No. 3, IMSOR, The Technical University of Denmark (1967).

[18] K. Spielberg, Algorithm for the simple plant location problem with some side conditions, *Operations Res.* 17 (1) (1969) 85–111.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 99–116

PARTIAL ORDERINGS IN IMPLICIT ENUMERATION

V. Joseph BOWMAN, Jr.

Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 15213, U.S.A.

James H. STARR

Bell Telephone Laboratories, Holmdel, NJ, U.S.A.

This paper investigates the use of general partial orderings in implicit enumeration algorithms. It is shown that if one chooses a partial order P such that xPy implies cx less than or equal to cy then there exists an optimal solution which is "prime" in the sense that the solution, x, is feasible and there exists no yPx such that y is feasible. Enumeration algorithms search for prime solutions and two methods of performing this search are characterized. Finally the paper illustrates these concepts by the introduction of two partial orders that are stronger than vector partial ordering which is the basis of Balas type implicit enumeration algorithms.

1. Introduction

In this paper we investigate partial orderings as they apply to implicit enumeration techniques for binary linear integer programs. This programming problem consists of linear constraints which define the set of binary solutions which may be considered, and a linear decision criteria or objective function which is used to find the optimal solution. Enumeration algorithms usually base their enumeration, at least implicitly, on some combination of objective function and feasibility considerations. The object is to find rules which generate a solution with a better objective function value than has yet been found, or if this cannot be done, to terminate. Orderings have been used by several authors as direct means of enumeration. Lexicographic orderings have been used by Dragan [6] and Korte, Krelle and Oberhofer [10]. Lawler and Bell [11] have used a combination of lexicographic and vector partial orderings. Balas [1] has used vector partial ordering as a basis for his additive algorithm.

We wish to investigate the explicit use of a partial ordering of binary solutions as a surrogate for the objective function. The motivation is that if one were provided a total order that corresponded to the ordering induced by the objective function, the programming problem is reduced to searching that linear ordering until a first feasible solution is found. It should be pointed out that the objective function does not totally order all solutions since several solutions may have the same objective value. In such cases we assume that these solutions of common value can be ordered in an arbitrary manner. This implies that if there are alternative optima we have no preference between them and are only interested in one of these solutions.

If we have a partial ordering that agrees with the objective function ordering, i.e., elements ordered in the partial order are ordered in the same direction in the objective order, then the partial order cannot contain as much information. In return for this loss, however, we may receive the following advantage: solutions that are potentially "better" may be easier to generate using the partial ordering rather than the objective function. This may be true especially if we choose a well structured partial order. For instance if c is the vector of costs and x and y are two binary solutions such that $x \leq y$, i.e., $x_i \leq y_i$ for all i, then $c \geq 0$ implies $cx \leq cy$. Thus if x is feasible, there is no need to evaluate y since it cannot be optimal if we are minimizing. Moreover we know that if w is an optimal solution then either $w \leq x$ or w is unordered with x. Thus we have implicitly enumerated all solutions y such that $x \leq y$. This ordering is the one used by the Balas additive algorithm.

In the next section we shall discuss properties of general partial orderings that agree with the objective function and how these properties can be used in theoretical enumeration algorithms. The third section discusses two partial orderings and their particular properties.

2. Partial orderings

We shall consider the problem

$$\begin{aligned} &\min \ cx, \\ &\ \text{s.t. } Ax \geq b, \\ &\qquad x \text{ binary}, \end{aligned} \tag{1}$$

where A is an $m \times n$ matrix and c, b and x are vectors of appropriate dimension.

We shall call any binary vector x a *solution* and if $Ax \geq b$ a *feasible solution*. Any feasible solution x such that $cy < cx$ implies y is not feasible is called an *optimal solution*.

Let P denote a partial ordering relationship on the solutions; that is for every two binary vectors x and y, $x \neq y$ only one of the following holds

(i) $x P y$,

(ii) $y P x$,

(iii) neither $x P y$ nor $y P x$,

and if $x P y$ and $y P z$, then $x P z$. One can think of P as a preference relationship; that is either x is preferred to y or y is preferred to x or neither of these. The vector partial ordering discussed in section 1 is generated by the relationship $P = \{\leq\}$; that is, P is the usual vector partial ordering relation.

If condition (iii) holds x and y are *unordered*. If $x P y$ ($y P x$) and there exists no z such that $x P z P y$ ($y P z P x$) we say that x and y are *adjacent*. A *chain* is a sequence of adjacent elements $(x^1, x^2, \ldots, x^k)$ with $x^i P x^{i+1}$ and such that there exists no y or z with $z P x^1$ and $x^k P y$. A *partial chain* is a sequence of adjacent elements $(x^1, \ldots, x^k)$ with $x^i P x^{i+1}$. x^1 is called the *origin* of the *partial chain*. We say that P^* *agrees with* P (P^* is contained in P) if $x P^* y$ implies $x P y$. If for all x, y binary either (i) or (ii) holds, then P generates a *complete* or *linear order*.

Throughout the remainder of this paper we shall consider only those partial order relationships such that $P \in C$ where

$$C = \{P \mid x P y \implies cx \leq cy \text{ for all } x, y \text{ binary}\}.$$

C^L will denote the set of linear orders in C. To simplify notation, the partial order generated by P will be called the partial order P.

The following definition identifies the central property of solutions in a partial order that is of concern in enumeration techniques.

Definition 1. A solution x is a *prime solution* with respect to a partial order P if
(i) x is feasible, and
(ii) if $y P x$ then y is infeasible.

We know that if x is prime then all chains through x cannot contain a feasible solution with lower cost, since all solutions y, such that $y P x$, are infeasible by definition and all solutions y, such that $x P y$, have $cx \leq cy$ since $P \in C$.

Lemma 1. *At least one optimal solution is a prime solution.*

Proof. Let y be an optimal solution such that $x P y$ implies x is not optimal. Now $x P y$ implies $cx \leq cy$ thus $x P y$ implies x infeasible since otherwise it is an alternative optima but y is feasible and $x P y$ implies x infeasible, thus y is prime.

Thus we can restrict our search to the prime solutions. Of course, the set of prime solutions change with different partial orders and thus in searching for partial orders one would desire to find a well-structured partial order with few prime solutions. It is the function of enumeration schemes to find the prime solutions and to identify them either directly or indirectly.

The distinction between direct and indirect enumeration techniques is not trivial. Each embodies a different enumeration technique that generates solutions in alternative ways. To contrast the differences we present a general algorithm for each.

Rudimentary Direct Algorithm:

Step 1: Choose a partial order $P \in C$.

Step 2: Generate a prime solution.

Step 3: Establish criteria that eliminate all chains containing this prime solution. Go to Step 2.

Because of the generality of this Algorithm, each of the steps is non-trivial. The selection of the partial order P in Step 1 must be chosen so that Step 2 can recognize when a prime solution is generated. Step 2 may contain a technique that always generates a prime solution or more practically may generate several solutions stopping when a prime solution is generated. Similarly Step 3 may be implemented in several ways. The direct implication is that chains are actually deleted from the partial order which is direct elimination of a set of solutions. A more practical way may be to impose additional constraints that make the present prime solution infeasible and that relies on multiple solution generation within Step 2. A direct search algorithm for set-covering problems has been used by Bowman and Starr [3] for a partial ordering that will be described in the next section.

The indirect algorithm is more familiar to students of implicit enumeration and is the search method used by Balas [1].

Rudimentary Indirect Algorithm:

Step 1: Choose a partial order $P \in C$ and a solution x such that there is no solution y with yPx.

Step 2: If x is feasible, go to Step 4. Otherwise go to Step 3.

Step 3: Choose a partial chain originating at x that contains a feasible solution. Generate the adjacent solution to x, say y, on this chain. Let $x = y$ and go to Step 2. If no such partial chain exists, go to Step 4.

Step 4: (Backtracking) Delete all partial chains originating at x. Retrace the current chain to x until there is an element w that is the origin for at least two chains. Let y be an adjacent element to w with wPy and such that y is not on the current chain. Let $x = y$ and go to Step 2. If there exists no such y terminate.

In the indirect algorithm, the choice of the partial order P is guided by the ease of finding adjacent elements (for Steps 3 and 4) and the ability to determine partial chains that contain a feasible solution. In actual practice Step 3 would probably be relaxed to the statement of finding a chain that has potential for a feasible solution; that is, it may not but we need to explore further. Step 4 implies the direct elimination of solutions as did Step 3 of the direct algorithm. Here again practical application would imply the addition of constraints that mark eliminated partial chains.

In order to better understand the implications of this partial ordering material and the two algorithms, we illustrate them with vector partial ordering, the ordering used by Balas for his additive algorithm. As noted earlier, this ordering is generated by the relationship $P = \{\leq\}$; that is, xPy if $x_i \leq y_i$ for all i. A graph of the partial order for $n = 4$ is shown in Figure 1. We have that $P \in C$ if $c \geq 0$ and since we can always replace a variable with $c_i < 0$ by its complement, i.e., $\bar{x}_i = 1 - x_i$, we have $c \geq 0$ for any problem (1). The smallest element in this chain is at the top of Figure 1 and as one follows any chain from top to bottom the objective function is monotone non-decreasing.

Let us investigate the Indirect Algorithm first as implemented by Balas [1]. We have chosen the partial order P and we now choose a starting solution $x_j = 0$ for all j. This completes Step 1. The criteria for adjacency is the setting of one variable

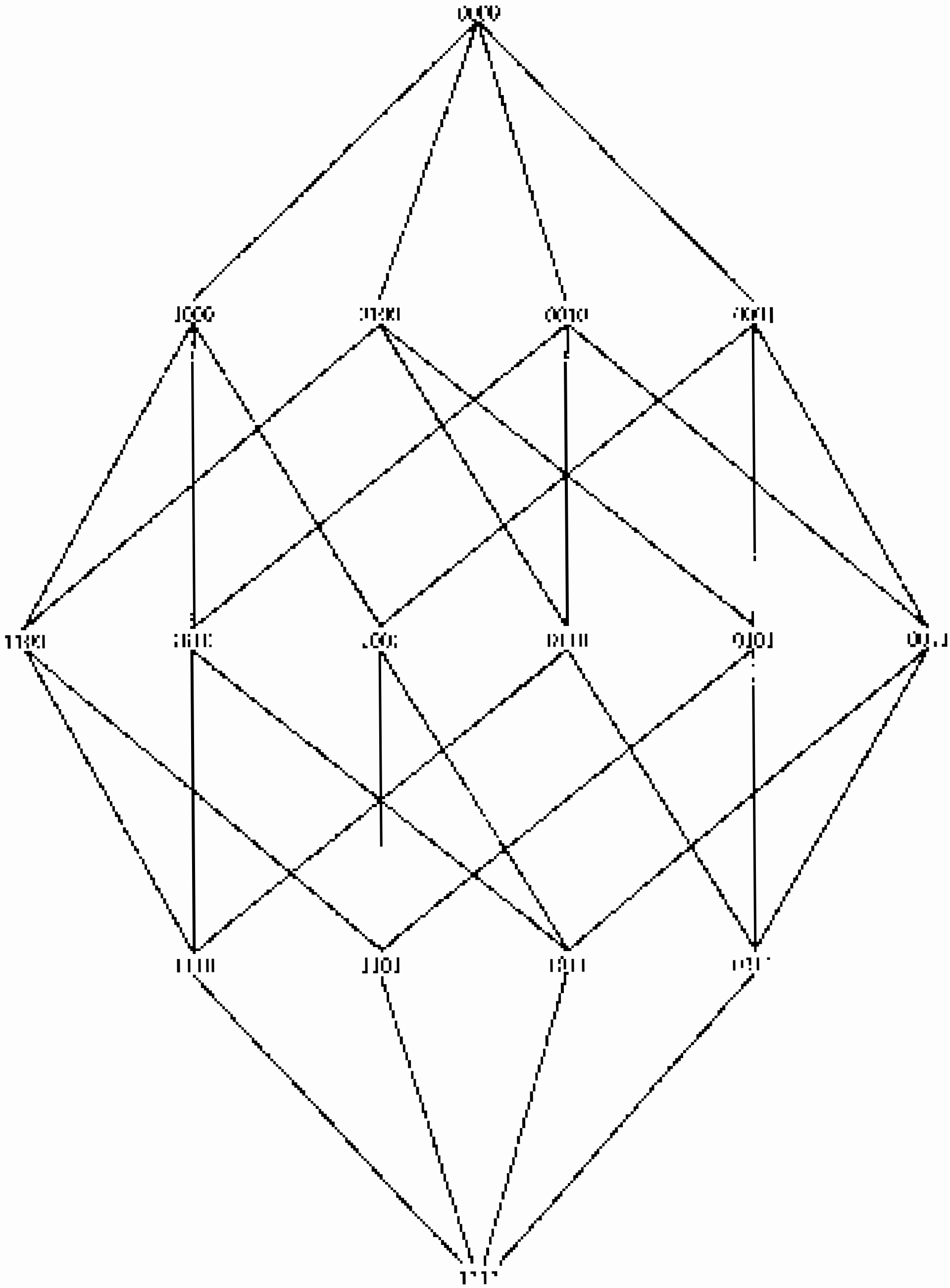


Fig. 1. Vector partial order, n = 4.

$x_j = 0$ to $x_j = 1$ or of setting $x_j = 1$ to $x_j = 0$. At each iteration of a Balas type algorithm the indices of variables are divided into two disjoint sets S and F. S denotes indices of variables that are *fixed* at a particular value. These variables constitute a *partial solution*. The remaining variables, those in index set F, are called *free*. In the tests for feasibility or optimality, these variables are implicitly assigned a value of zero. Thus each iteration corresponds to a solution of the problem. The algorithm at a solution (or iteration) that is not feasible investigates by various criteria (see for example [1, 2, 7, 8, 9, 12]), to see if setting a variable with index in F to one may lead to a feasible solution. If so the variable is set to one and its index added to S and deleted from F. If some variable with index in F cannot yield a feasible solution with value one it is fixed at zero, put in S and deleted from F and is marked so that setting it to one from this solution will not be attempted. This marking process corresponds to a partial chain elimination. It says that from this solution any chain containing the marked index has been eliminated with the variable at its other value. This corresponds to Step 3. The backtracking in Step 4 implicitly says that any partial chain *originating* at the present solution cannot contain an optimal solution. The backtracking frees variables in S in the reverse order they were added and stops at the first unmarked index, say k. Variable k is then set at its opposite value and marked indicating that all partial chains originating at its former value have been eliminated. This is Step 4.

Consider the following example:

Example 1. The problem is

$$\min\ x_1 + 2x_2 - 3x_3 + 4x_4,$$

$$\text{s.t. } x_1 + 3x_2 - x_3 \geq 2,$$

$$2x_1 + x_2 + x_3 + x_4 \geq 1.$$

A sequence of solutions that might be generated by an Indirect Algorithm is displayed in Table 1.

It should be noted that several of the sequence numbers are generated by one step of the indirect algorithm. In practice one iteration of an algorithm would also generate several of these sequence numbers. They are explicitly stated here to emphasize the investigative properties of the Indirect Algorithm. Reference to Figure 1 may aid in understanding the sequence of exploration and the deletion of chains.

As mentioned earlier the use of a Direct Algorithm has been limited to another partial ordering discussed by Bowman and Starr. It is therefore necessary to discuss an algorithm that will first find a prime solution and second indicate what solutions are not contained in the chains containing this prime solution for vector partial ordering. These are described below.

Prime Solution Generation:

Step 1: Set iteration counter $t = 1$ and $j_0 = n - 1$, $b_0 = b$.

Table 1.

Sequence		Solution	Information
0	Step 1	(0000)	infeasible
1	Step 3	(1000)	infeasible
2	Step 3	(1100)	feasible
3	Step 4	(1000)	eliminate all partial chains originating at (1100)
4		(1010)	non-optimal
5	Step 4	(1000)	eliminate all partial chains originating at (1010)
6		(1001)	non-optimal
7		(1000)	eliminate all partial chains originating at (1001)
8	Step 4	(0000)	eliminate all partial chains originating at (1000)
9		(0100)	infeasible
10	Step 3	(0110)	non-optimal
11	Step 4	(0100)	eliminate all partial chains originating at (0110)
12		(0101)	non-optimal
13		(0100)	eliminate all partial chains originating at (0101)
14	Step 4	(0000)	eliminate all partial chains originating at (0100)
15		(0010)	non-optimal
16	Step 4	(0000)	eliminate all partial chains originating at (0010)
17		(0001)	non-optimal
18	Step 4	(0000)	eliminate all partial chains originating at (0001)

Step 2: Find k such that

$$\sum_{r=1}^{k} (a_{ir})^+ \geq b_{ir} \quad \text{for all } i,$$

$$\sum_{r=1}^{k-1} (a_{ir})^+ < b_{ir} \quad \text{for some } i,$$

where $(a_{ir})^+ = \max(a_{ir}, 0)$. If $k < j_{r-1}$, go to Step 3. Otherwise go to Step 4.

Step 3: Set $j_r = k$, $b_{i,r+1} = b_{ir} - a_{ik}$ for all i. If $b_{i,r+1} \leq 0$, stop. Otherwise let $r = r + 1$ go to Step 2.

Step 4: Set $j_{r-1} = j_{r-1} + 1$, $k = j_r$, $b_{ir} = b_{ir} - a_{ik} + a_{i,k+1}$ for all i. If $j_{r-1} \neq j_{r-2}$ go to Step 2. Otherwise go to Step 5.

Step 5: Set $r = r - 1$. If $r = 0$, stop no prime solution. Otherwise go to Step 4.

This algorithm is a systematic way of exploring the prime covers and a version of it is used by Bowman and Starr [3] for set covering problems.

To obtain solutions that are unordered with a prime solution, say y, any other solution x must satisfy the following two constraints:

$$\sum_{j \in O} x_j \geq 1, \qquad \sum_{j \in R} x_j \leq |R| - 1 \tag{1}$$

where $O = \{j \mid y_j = 0\}$ and $R = \{j \mid y_j = 1\}$. That is, a solution, x, is unordered with y if and only if a zero value in y becomes one and a one value in y becomes zero. We will combine these two by adding the constraints of (1) to the problem (I) after every prime solution is generated.

Example 2. Consider the problem of Example 1 and the algorithm described above. The sequence of solutions and constraints generated are as in Table 3.

Table 3

Sequence	Solution	Constraints	
1	(1100)	$x_1 + x_2 \geq 1$	$x_3 + x_4 \geq 1$
2	(1010)	$x_1 + x_3 \geq 1$	$x_2 + x_4 \geq 1$
3	(1001)	$x_1 + x_4 \geq 1$	$x_2 + x_3 \geq 1$
4	(0111)	$x_2 + x_3 + x_4 \geq 2$	$x_1 \geq 1$

In this case very few solutions are generated, however, the computation of the prime solutions is not trivial and comprises the bulk of computation. We know of no attempts to implement such an algorithm. The version presented here would most certainly perform poorly with respect to sophisticated versions of a Balas Indirect Algorithm and would require in depth research to find efficient means of generating prime solutions.

The approach to these two algorithms is completely different. The direct algorithm requires efficient means of generating prime solutions and means of eliminating all chains through a prime solution. The Indirect Algorithm needs efficient means of finding adjacent solutions, detecting infeasibility in partial chains and efficient means of eliminating partial chains. Since the requirements for these algorithms are different it may be the case that different partial orders would respond better to one algorithm than the other. However before such investigations can be undertaken it is important to generate other partial orders than the vector partial ordering. The next section discusses two such orderings and illustrates the reductions that take place in terms of the number of chains.

3. Two specific partial orders

The first partial ordering we wish to consider has been discussed by Bowman and Starr [3, 4] for the set covering problem, by Starr [15] for the 0–1 problem where $A \geq 0$, and by Gale [16] and Zimmerman [17] with respect to Matroids. In the Bowman and Starr papers the ordering relationship P was represented by $\{\ll\}$, and in the Zimmerman paper P was represented as $\ll$. In this paper we will use P^2 to denote this ordering. The two comes from the relationship of the $\ll$-ordering to two comparability in switching functions. This relationship is discussed by Bowman and Starr in [5] and for reference to twocomparability the reader should refer to Muroga [13]. In the same manner we can refer to the vector partial ordering as P^1 since it corresponds to 1-comparability. This numbering is also significant in that P^1 agrees with P^2, i.e., if xP^1y then xP^2y.

The P^2 ordering is defined from the vector partial ordering of the *index vectors* of binary vectors.

Definition 2. The *index vector*, $s(x)$, of a binary solution x has the following properties, where x_0 is the number of zero components of x.

(a) $s_1(x) = s_2(x) = \cdots = s_{x_0}(x) = 0$,

(b) $s_{x_0+1}(x) < s_{x_0+2}(x) < \cdots < s_n(x)$,

(c) $x_i = 1$ if and only if there exists a j such that $s_j = i$.

The index vector lists the positive indices of a binary vector in increasing order.

Definition 3. Let x and y be any two solutions with associated index vectors $s(x)$ and $s(y)$. If $s(x) \leq s(y)$, then $x P^2 y$. Here "$\leq$" denotes the usual vector partial ordering.

The P^2 ordering defined by the index vectors satisfies our requirement for decreasing desireability moving down a complete chain if the objective function is linear and the elements of the objective function form a monotone non-decreasing sequence.

The following lemma is proven in [3]:

Lemma 2. *If $0 \leq c_1 \leq c_2 \leq \cdots \leq c_n$ and $x P^2 y$, then $cx \leq cy$.*

Lemma 3, proven in [3], shows that the vector partial ordering, P^1, usually used in programming algorithms, agrees with the P^2 ordering.

Lemma 3. *If $x \leq y$, then $x P^2 y$.*

The vector partial ordering is used as the basis for implicit enumeration, which may be regarded as the enumeration of complete chains in the graph induced by this ordering. Since the vector partial order agrees with the P^2 ordering, the P^2 ordering must have fewer complete chains.

Figure 2 shows the P^2 ordering for $n = 4$. It is important to note that every implication in Figure 1 is contained in Figure 2; this is from Lemma 3. In addition, the number of prime solutions will be smaller in P^2 than in P^1 because of the existence of additional relationships. In particular, note for the problem in Example 1 that three of the prime solutions under the P^1 ordering are not prime under the P^2 ordering; that is, the solutions (1010), (1001) and (0111) are all ordered with (1100) in the P^2 ordering. Bowman and Starr [3] provide a method for generating successive prime solutions on this ordering for the set covering problem. This generation is exploratory in nature in that several non-prime solutions may be generated to find a prime solution. However, the computation times presented have been quite small especially for those problems with distinct costs.

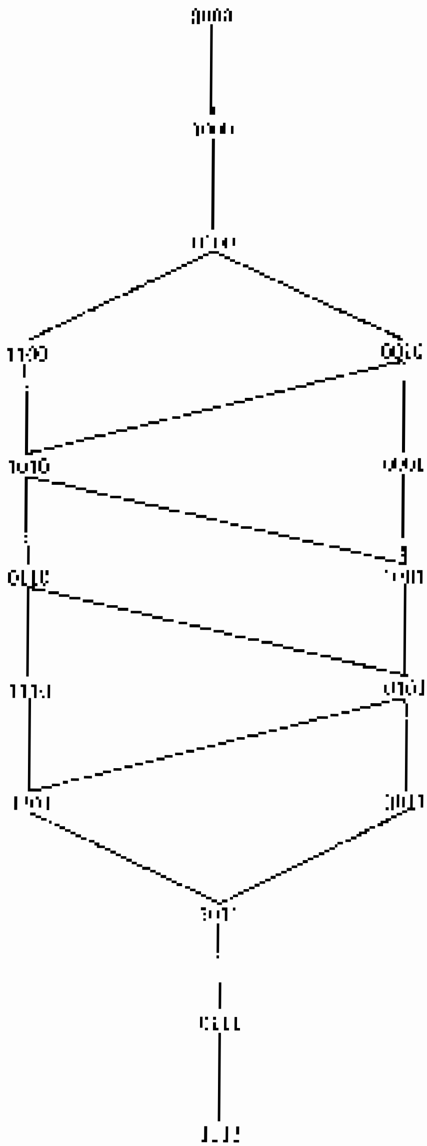


Fig. 2. P'-order $n = 4$.

The following examples highlight the gains that can be made by using the P' ordering.

Example 3. Consider again the problem of example 1. If one uses a Direct Algorithm the sequence of solutions would be as in Table 3.

Table 3.

Sequence	Prime Solution	Comments
1	(1100)	only (0010) and (0001) are considered
2	Stop	neither (0010) or (0001) is feasible

Example 4. An Indirect Algorithm would generate the sequence in Table 4.

Table 4

Sequence	Solution	Comments
0	(0000)	infeasible
1	(1000)	infeasible
2	(0100)	infeasible
3	(1100)	feasible
4	(0100)	eliminate all partial solutions originating at (1100)
5	(0010)	non-optimal
6	(0100)	eliminate all partial solutions originating at (0010)
7	(1000)	eliminate all partial solutions originating at (0100)
8	(0000)	eliminate all partial solutions originating at (1000)

In both these examples no specific way is described of eliminating the partial chains or of finding prime solutions. In fact there does not exist at present an indirect algorithm for exploring this ordering outside of the very rudimentary one described above. However, it is important to note that this ordering has the significant effect of reducing the number of solutions that must be generated by either a direct or indirect algorithm. This is offset however by the need for more complicated algorithms to generate the successive solutions. In light of recent research by Piper [14], showing that logical tests in Balas type enumerations have strong influence on computation time, it might be suspected that efficient use of the P^2 ordering would be helpful in enumeration techniques. This has been supported by the success of the Bowman and Starr algorithm for set-covering.

There exists a further refinement of the P^2 ordering that has not been investigated by any authors. The P^2 ordering required the knowledge of the ordering of the cost coefficients. This new ordering, which we denote P^3, requires that the cost coefficients be positive and in non-decreasing order and that one also knows the ordering of the first differences of the cost coefficients.

Let $\Delta_i = c_i - c_{i-1}$, where $c_0 = 0$. Then we have

$$\sum_{k=1}^{n}\left(\sum_{i=k}^{n} x_i\right)\Delta_k = \sum_{i=1}^{n} c_i x_i. \tag{2}$$

We now define the P^3 ordering by the *difference vector* $d(x)$ of a binary vector x

Definition 4. The *difference vector*, $d(x)$, of a binary vector x has the following properties:

(a) $d(x)$ has $n(n+1)/2$ components.

(b) $0 \le d_1(x) \le d_2(x) \le \cdots \le d_{n(n+1)/2}(x)$.

(c) Let $\{j_1, j_2, \ldots, j_n\}$ be a permutation of $\{1, 2, \ldots, n\}$. Then $\sum_{i=k}^{n} x_i$ components of d have value p if $j_p = k$.

The importance of the difference vector lies in the creation of the cost function in terms of first differences. This transformation is

$$\sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n(n+1)/2} \Delta_{j_{d_i(x)}}$$

and follows directly from (7) and part (c) of the definition.

The difference vector involves two index sets, the index set $\{1, \ldots, n\}$ on the elements of x and a permutation of these, $\{j_1, j_2, \ldots, j_n\}$ associated with the first differences of c. The following example shows the construction of a difference vector:

Example 5. Assume $n = 4$, $j_1 = 2$, $j_2 = 1$, $j_3 = 4$, $j_4 = 3$. If $x = (1010)$ then since $\sum_{i=4}^{4} x_i = 0$, 0 components of d have value 3 because $j_3 = 4$; since $\sum_{i=3}^{4} x_i = 1$, 1 component of d has value 4 because $j_4 = 3$; since $\sum_{i=2}^{4} x_i = 1$, 1 component of d has value 1 because $j_1 = 2$; and since $\sum_{i=1}^{4} x_i = 2$, 2 components of d have value 2 because $j_2 = 1$. Thus $d(x) = (0, 0, 0, 0, 0, 0, 1, 2, 2, 4)$.

Furthermore,

$$\begin{aligned} cx &= \Delta_{j_1} + \Delta_{j_2} + \Delta_{j_2} + \Delta_{j_4} \\ &= \Delta_2 + \Delta_1 + \Delta_1 + \Delta_3 \\ &= (c_2 - c_1) + c_1 + c_1 + (c_3 - c_2) \\ &= c_3 + c_1. \end{aligned}$$

Definition 5. Let x and y be any two solutions with associated difference vectors $d(x)$ and $d(y)$. If $d(x) \le d(y)$, then $x P^2 y$.

The P^2 ordering satisfies our requirement for decreasing desirability if the objective function is linear, the elements of the objective form a monotone non-decreasing sequence and the first differences are monotone non-decreasing on the index set $\{j_1, j_2, \ldots, j_n\}$.

Lemma 4. *If $0 \le c_1 \le c_2 \le \cdots \le c_n$, $0 \le \Delta_{j_1} \le \Delta_{j_2} \le \cdots \le \Delta_{j_n}$ and $x P^2 y$, then $cx \le cy$.*

Proof. Assume the ordering relationships and $x P^2 y$. We have

$$cx = \sum_{i=1}^{n(n+1)/2} \Delta_{j_{d_i(x)}}$$

Since $x P^3 y$, $d(x) \leq d(y)$ and by the ordering on first differences $\Delta_{i_j(x)} \leq \Delta_{i_j(y)}$. Thus

$$cx \leq \sum_{j=1}^{n(x)} \Delta_{i_j(y)} = cy.$$

This proves the lemma.

Similar to the relationship of the P^1 and P^2 ordering we find that the P^2 ordering agrees with the P^3 ordering.

Lemma 5. *If $x P^2 y$, then $x P^3 y$.*

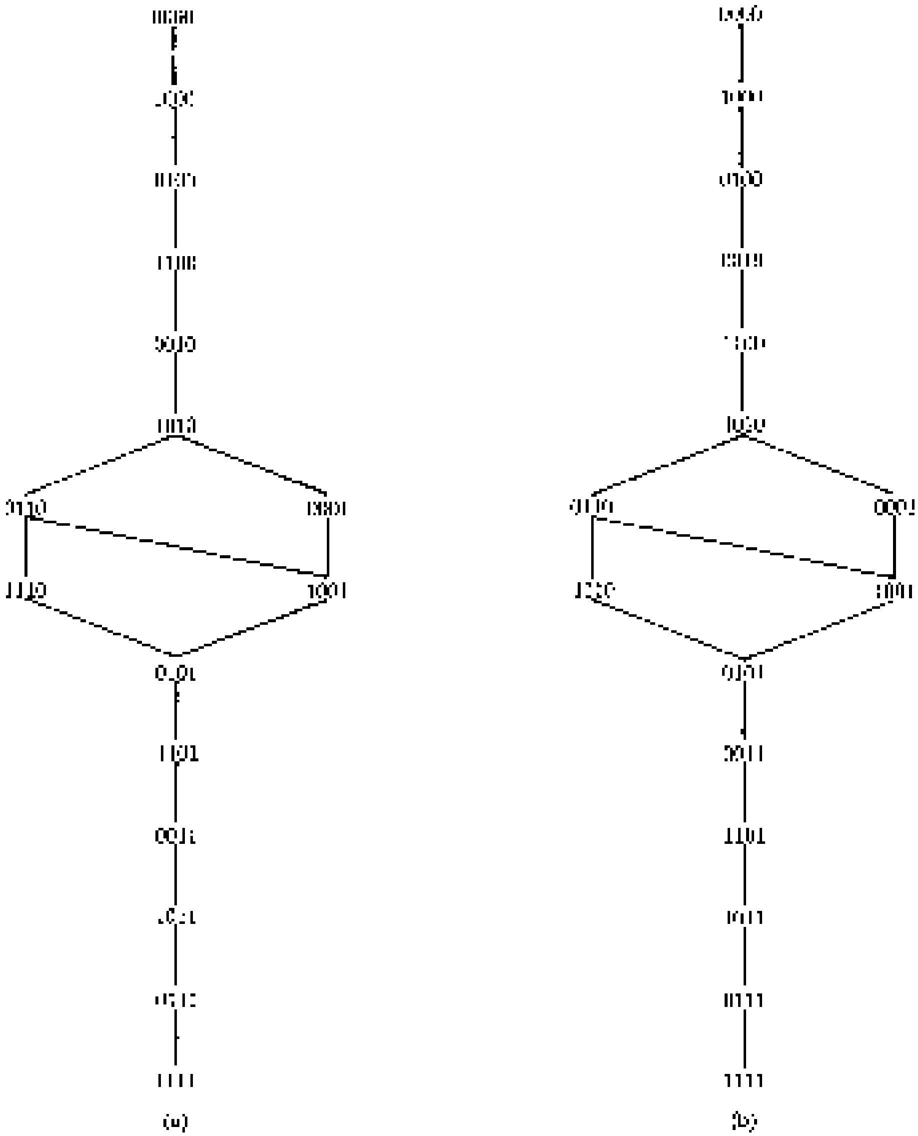
Proof. If $x P^2 y$ then $s(x) \leq s(y)$. Recall these are index vectors. By Definition 4 we have

$$\sum_{i=s_k(x)}^{n} x_i = n - k + 1, \qquad \sum_{i=s_k(y)}^{n} y_i = n - k + 1.$$

In addition since $s_k(x) \leq s_k(y)$ we have $\sum_{i=s_k(x)}^{n} y_i \geq n - k + 1$. Since the elements of $d(x)$ and $d(y)$ are monotone non-decreasing it immediately follows that $d(x) \leq d(y)$.

In the same manner that P^2 contains more information than P^1, this lemma shows that P^3 contains more information than P^2. However, this new information is at the cost of a more complicated ordering vector (compare the generation of $s(x)$ to the generation of $d(x)$). Moreover while for each n, P^1 and P^2 generate just one ordering, P^3 generates several orderings for a given n because of various relationships on the first differences. For the case with $n = 4$ there are 8 different partial orders. These are shown in Fig. 3a through 3h along with the orderings on the first differences that generate them. It is interesting to note that four of the orders are linear orders. This is important since either a Direct or Indirect Algorithm would terminate with the generation of the first feasible solution on these orders.

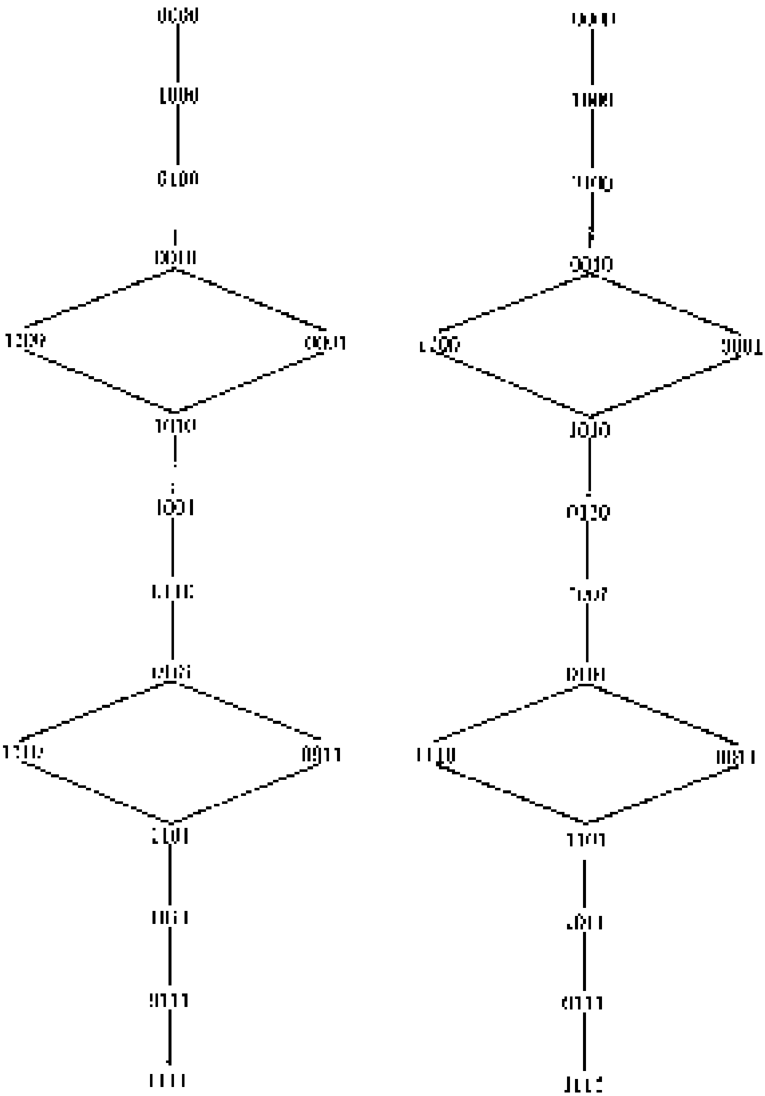
Remark. There are no known algorithms for searching the P^3 ordering. In fact it is not clear that the representation of this order by the difference vector $d(x)$ is the most efficient method. It does give a means for beginning research on the importance of this ordering in enumeration techniques. This is best exemplified by the problem of Example 1. In this problem $\Delta_i = 1$ $i = 1, 2, 3, 4$ and consequently we could choose any of the 4! orderings of the first differences. If we look at all the orderings we find that both a Direct and Indirect Algorithm will terminate with the generation of the first feasible solution for all orders except those shown in Fig. 3c and 3d. This is because all elements in the other orders are ordered with the solution (1100). For the orderings shown in Fig. 3c and 3d, the Direct Algorithm will generate one prime solution as in Example 3, while an indirect algorithm will backtrack and go forward only once, this latter being the examination of (1100) and (0001).



P^1-order, $n = 4$,
$\Delta_1 \prec \Delta_2 \prec \Delta_3 \prec \Delta_4$,
[illegible]
[illegible]
[illegible]
[illegible]

P-order, $n = 4$,
[illegible]
[illegible]
[illegible]

Fig. 3.



(c)

(d)

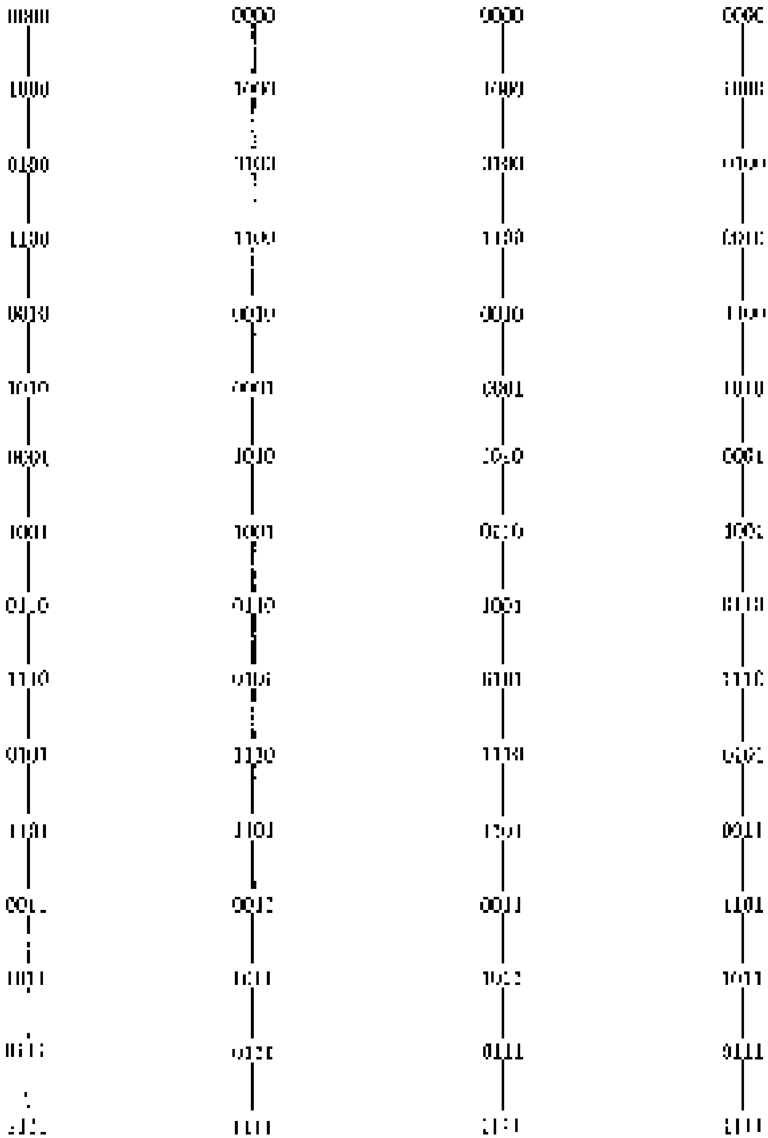
P-order, $n = 4$.

[illegible]

P-order, $n = 4$.

[illegible]

Fig. 3(cont.)



(e)	(f)	(g)	(h)
P'-order, $n = 4$, [illegible] [illegible] [illegible]	P'-Order, $n = 4$, [illegible] [illegible] [illegible]	P'-order, $n = 4$, [illegible]	P'-Order, $n = 4$, [illegible]

Fig. 7(cont.)

Summary

This paper has been an exposition on partial orders in enumeration algorithms. The exploitation of partial orders gives rise to two types of algorithms, one which directly searches for prime solutions and one which indirectly searches for prime solutions. To-date most work in implicit enumeration has dealt with indirect algorithms applied to vector partial ordering. This paper has described a rudimentary direct algorithm for vector partial ordering. In addition it has described two other partial orderings that are successive incorporation of more cost information. Only a direct algorithm has been developed for the P^2 ordering and has demonstrated some computational success for set-covering problems. This paper, however, has not closed any doors on enumeration techniques. Instead it has raised many questions and many additional directions for further research in enumerative methos.

Some of the questions raised are:

(a) What type of partial orderings should be explored?

(b) Do Indirect Algorithms always dominate Direct Algorithms or vice-versa?

(c) Is there a computationally efficient Direct Algorithm for vector partial ordering?

(d) Are there computationally efficient Direct and Indirect Algorithms for the P^2 and P^3 partial orders?

It is obvious that these questions can be answered only by further research. It is also obvious that the natural nesting of the P^1, P^2 and P^3 orderings can be expanded until a complete ordering is generated. Moreover, it is not clear that this sequence of partial orders is in any way the "best". At a time when implicit enumeration has shown its worth and when most researchers are exploring refinements on vector partial orderings these ideas raise a whole new avenue to pursue in the area of enumeration.

5. Acknowledgement

We would like to thank an anonymous referee for his useful suggestions.

References

[1] E. Balas, An additive algorithm for solving linear programs with zero-one variables, *Operations Res.* 13(4) (1965) 517–546.

[2] E. Balas, Discrete programming by the filter method, *Operations Res.* 15(5) (1967) 915–957.

[3] V.J. Bowman and J.H. Starr, Set covering by ordinal cuts I: linear objective functions, Working Paper 97-72-3, Graduate School of Industrial Administration, Carnegie-Mellon University, June, 1973.

[4] V.J. Bowman and J.H. Starr, Set covering by ordinal cuts II: partial preference functions, Working Paper 10-73-4, Graduate School of Industrial Administration, Carnegie-Mellon University, July, 1973.

[5] V.J. Bowman and J.H. Starr, Two-comparable prime implicants and canonical switching functions, Management Sciences Research Report 338, Graduate School of Industrial Administration, Carnegie-Mellon University, January 1974.
[6] L. Dragan, Un algorithme lexicographique pour la résolution des programmes linéaires en variables binaires, *Management Sci. Theory* 16 (1969) 246–252.
[7] N. Driebeek, An algorithm for the solution of mixed integer programming problems, *Management Sci.* 12(7) (1966) 576–587.
[8] A. Geoffrion, An improved implicit enumeration approach for integer programming, *Operations Res.* 17(3) (1969) 437–454.
[9] F. Glover, A multiphase-dual algorithm for the zero-one integer programming problem, *Operations Res.* 13(6) (1965) 879–919.
[10] B. Korte, W. Krelle and W. Oberhofer: Ein lexikographischer Suchalgorithmus zur Lösung allgemeiner ganzzahliger Programmierungsaufgaben, *Unternehmensforschung*, Part 1: *13* (1969) 73–98 Part 2: *13* (1969) 171–192.
[11] E.L. Lawler and M.D. Bell, A method for solving discrete optimization problems, *Operations Res.* 14 (1966) 1098–1112.
[12] C. Lemke and K. Spielberg, Direct search algorithms for zero-one and mixed-integer programming, *Operations Res.* 15(5) (1967) 892–914.
[13] S. Muroga, *Threshold Logic and its Applications* (Wiley, New York, 1971).
[14] C. Piper, Computational studies in optimizing and feasibility-testing linear programs in zero-one variables, Ph.D. Thesis, Carnegie-Mellon University, 1975.
[15] J.H. Starr, Zero-one programming: A partial ordering and its use in the optimization of numerical and non-numerical functions, Ph.D. Thesis, Carnegie-Mellon University, 1975.
[16] D. Gale, Optimal assignments in an ordered set: An application of matroid theory, *J. Combinatorial Theory* 4 (1968) 176–180.
[17] U. Zimmermann, Some partial orders related to boolean optimization and the greedy algorithm, *Ann. Discrete Math.* 1 (1977) 539–550.

Annals of Discrete Mathematics 1 (1977) 117–143
© North-Holland Publishing Company

A SUBADDITIVE APPROACH TO SOLVE LINEAR INTEGER PROGRAMS

Claude-Alain BURDET
Systemathica Consulting Group, Pittsburgh, PA 15213, U.S.A.

Ellis L. JOHNSON
IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

A method is presented for solving pure integer programs by a subadditive method. This work extends to the integer linear problem a method for solving the group problem. It uses some elements of both enumeration and cutting plane theory in a unified setting. The method generates a subadditive function and solves the original integer linear program.

1. Introduction

In a previous paper [5], we have developed an algorithm to solve the group problem derived from an integer linear program using an approach based on constructing a subadditive diamond gauge yielding a valid inequality. The group problem, however, represents a *relaxation* of the original integer linear program in that the optimal group solution need not be feasible with respect to all the initial linear programming constraints. We now present an extension of our subadditive method which will solve the original integer linear programming problem (ILP): the group structure on the one hand and *all* the constraints of the initial linear program on the other are both taken into account in the following developments.

The approach here is fundamentally different from branch-and-bound. One difference is that we only keep one problem rather than dividing it up into subproblems. Nor is the method like existing cutting plane methods. It begins by adjoining an initial set of Gomory mixed integer cuts to the tableau, and solving the resulting augmented linear program. However, no more cuts are generated explicitly. Instead, a group enumerative phase is entered, and subsequently the method alternates between group enumeration and parameter adjustment via a linear programming (LP) problem. In the enumerative phase, we do not generate an enumeration tree. Instead, an LP column is generated for each enumerated point, and a better subadditive function is generated by parameter adjustment. Despite a surface similarity, the method does not resemble enumerative cuts [3] because the enumeration is not used to explicitly derive cuts which exclude part of the linear programming feasible region. In fact, our enumerated points will often be outside of the linear programming feasible region and will still help give progress.

The approach here resembles the duality methods of Fisher, Northup, and Shapiro [6] more than any other work. To illustrate this point, one could say that the "dual" problem here is based on the following program:

$$\max \pi_0,$$

$$\pi_0 \leq \pi(x),$$

$$c_j \geq \pi(\delta^j), \; \delta_i^j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}, \; j = 1, \ldots, n,$$

for all x satisfying some integer programming restrictions (the specific form used here will be (2), (3), and (4) below); for a subadditive function π on $\mathbf{R}^n$; that is,

$$\pi(x^1 + x^2) \leq \pi(x^1) + \pi(x^2).$$

The difficulty with this "dual problem" is that evaluating $\pi(x)$ for all x feasible to an integer program is as hard as solving the original integer program. The attempt here is to relax the restriction on x satisfying the integer programming restrictions while keeping the desirable properties of a dual:

π_0 is a lower bound on the objective function z,

$\max \pi_0 = \min z$.

Our dual problem will involve several components:

(i) the enumerated set X_E (Section 4);

(ii) various types of relaxation of (2), (3), and (4) which will be realized by the choice of S (Section 2);

(iii) parameters $\gamma^+, \gamma^-, \alpha^+, \alpha^-$, which enter in the definition of π and Δ (Section 3).

The problem (29), (30), (31) in Section 7 is of this form. For a given X_E, there may indeed exist a "duality gap"; that is, $\max \pi_0 < \min z$, even for the best γ's and α's. However, adequate enumeration will eventually cause this gap to disappear; typically, then, an integer feasible point x enters X_E and the problem is solved. It can happen that when an integer feasible x enters X_E, it may not be optimal, or it may be optimal but cannot be proven optimal.

Finally, let us mention another interesting approach due to Bell [2]. He closes the duality gap by a "supergroup" approach, which embeds the original group problem in a larger group.

1.1. The group problem

The derivation of the group problem from a linear integer program is well-known [7]. The group problem can be stated as follows:

$$\text{minimize } z = \sum_{j=1}^{n} c_j x_j, \tag{1}$$

$$\sum_{j=1}^{n} g_{ij}x_j \equiv g_{i0} \pmod 1, \; i = 1, \ldots, m_1, \tag{2}$$

$$x_j \geq 0 \quad \text{and integer}, \; j = 1, \ldots, n. \tag{3}$$

Denoting the columns of the matrix $G = (g_{ij})$ by g^j, (2) can be restated as

$$Gx = \sum_{j=1}^{n} g^j x_j \equiv g^0 \pmod 1.$$

To clarify the various concepts introduced, we will illustrate them with the following small example. The initial integer program is:

$$\begin{aligned} \text{minimize} \quad & 9x_1 + 23x_2 + 10x_3 \\ \text{subject to} \quad & 4x_1 + 11x_2 + 5x_3 = 12, \\ & x_j \geq 0 \text{ and integer.} \end{aligned}$$

The resulting group constraint is

$$\tfrac{4}{5}x_1 + \tfrac{1}{5}x_2 \equiv \tfrac{2}{5} \pmod 1,$$

$$x_1, x_2 \geq 0 \text{ and integer.}$$

1.2. *The constrained group problem*

In formulating the group problem (1)–(3), the non-binding constraints at the linear programming optimum are dropped. Such constraints can all be expressed as linear inequality constraints on the current non-basic variables $x_1, \ldots, x_n$. In order to restore the full set of original LP constraints, we shall consider here a general form of the problem where the constraints (2) and (3) are taken together with a set of linear inequalities (4):

$$\sum_{j=1}^{n} a_{ij}x_j \geq a_{i0}, \; i = 1, \ldots, m_2, \tag{4}$$

or, with $A = (a_{ij})$,

$$Ax = \sum_{j=1}^{n} a^j x_j \geq a^0.$$

Throughout, a^j and g^j will denote the jth columns of the matrices $A = (a_{ij})$ and $G = (g_{ij})$ in (4) and (2), respectively.

The updated linear programming constraints will be used here as the inequality (4). In our example above, it can be written as

$$-4x_1 - 11x_2 \geq -12.$$

The resulting constrained group problem is, thus,

$$\text{minimize } z = x_1 - x_2,$$

$$\tfrac{4}{5}x_1 + \tfrac{1}{5}x_2 \equiv \tfrac{2}{5} \pmod 1,$$

$$-4x_1 - 11x_2 \geq -12,$$

$$x_1, x_2 \geq 0 \text{ and integer,}$$

which is equivalent to the original integer program (with $x_3 = (12 - 4x_1 - 11x_2)/5$).

In general the system (4) will simply consist of those constraints of the original ILP which belong to the basic variables at the LP optimum (i.e. non-binding constraints); but one may also consider adjoining some new additional constraints. In any case, it should be apparent that this framework allows one to choose a system of LP constraints (4) so that any solution to (2), (3) and (4) satisfies all the constraints of the pure integer linear program. Naturally, when the group solution already satisfies all of the non-binding constraints, the restriction (4) will be satisfied by the optimum solution to (1), (2), (3) and need not be used at all.

As for our previous method [5] for the group problem, the present approach does not require the explicit determination of the group structure (via the Smith normal form, for example), nor is it critically dependent upon the order of the group which may be very large in practice. In this method the group structure $\mathcal{G}$ is read directly from the system (2) which comes from the updated linear programming tableau; in fact, $\mathcal{G}$ is merely considered here as a subgroup of the group structure I^n of infinite order [10] implied by the integrality requirements (3).

2. Valid inequalities

Definition 1. The inequality

$$\sum_{j=1}^{n} \pi_j x_j \geq \pi_0$$

is called *valid* if it is satisfied by all $x_1, \ldots, x_n$ satisfying (2), (3), and (4).

Definition 2. Given the constraints (2), (3), (4), define the following sets for each $x_1, \ldots, x_n$ with $x_j \geq 0$ and integer:

$$S_I(x) = \{y \mid y \geq x \text{ and (2), (3) and (4) hold for } y\},$$

$$S_G(x) = \{y \mid y \geq x \text{ and (2) and (3) hold for } y\},$$

$$S_L(x) = \{y \mid y \geq x \text{ and (4) holds for } y\}.$$

Note that the ILP, group, and LP feasible sets can be denoted by $S_I(0)$, $S_G(0)$, and $S_L(0)$.

In our example from Section 1.2,

$$S_I(0) = \{(3,0)\},$$

$$S_G(0) = \{(0,2) + (5k_1, 5k_2), (3,0) + (5k_1, 5k_2)\},$$

for all $k_1, k_2 \geq 0$ and integer,

$$S_L(0) = \text{the LP feasible region.}$$

For each of these S sets, we sometimes consider the points

$$(u, \xi) = (Gy, Ay) \in \mathbf{R}^{m_1} \times \mathbf{R}^{m_2}, \quad y \in S.$$

Definition 3. Define the *integer, group,* and *linear subpath sets* as

$$X_I = \{x \geq 0, \text{ integer} \mid S_I(x) \neq \emptyset\},$$

$$X_G = \{x \geq 0, \text{ integer} \mid S_G(x) \neq \emptyset\},$$

$$X_L = \{x \geq 0, \text{ integer} \mid S_L(x) \neq \emptyset\},$$

respectively.

For the example, we have

$$X_I = \{(0,0), (1,0), (2,0), (3,0)\},$$

$$X_G = \{(k_1, k_2) \mid k_1, k_2 \geq 0 \text{ are integer}\},$$

$$X_L = \{(0,0), (1,0), (2,0), (3,0), (0,1)\}.$$

These sets are called subpath sets because for $x \in X_I$, for example, there exists $y \in S_I(x)$, and if we think of y as generating a path using edges (G^j, A^j) from the origin to a point

$$(Gy, Ay) \quad \text{with } Gy = g^0, Ay \geq a^0,$$

then $x \leq y$ generates a subpath of that path.

Definition 4. For a given $S \subseteq \mathbf{R}^n$, define the *subclosure* X of S to be

$$X = \{x \geq 0, \text{ integer} \mid x \leq y \text{ for some } y \in S\}.$$

For $Y \subseteq \mathbf{Z}^n$, define Y to be *subinclusive* if the subclosure of Y is equal to Y.

With this definition, X_I, X_G, X_L are the subclosures of $S_I(0)$, $S_G(0)$, and $S_L(0)$, respectively. In the above definition of subclosure, S need not be contained in $\mathbf{Z}^n$, but the subclosure X is contained in $\mathbf{Z}^n$. In particular, $S_L(0)$ has non-integer points in it.

We summarize with a property below.

Property P1. (a) *The sets X_I, X_G, X_L are, respectively, the subclosures of $S_I(0)$, $S_G(0)$, $S_L(0)$;*

(b) *each of the sets X_I, X_G, X_L is subinclusive.*

This property and the next one are illustrated by the preceding example.

Proof. (a) If $x \in X_I$, then there exists $y \geq x$, $y \in S_I(0)$, by definition of X_I. Hence, x belongs to the subclosure of $S_I(0)$, and since X_I is defined to be precisely such x, the subclosure of $S_I(0)$ is X_I. The proof for I replaced by G or L is similar.

(b) This property follows from the fact that the subclosure X of any set S is subinclusive.

Property P2. (a) *$S_I(x) \subseteq S_G(x)$ and $S_I(x) \subseteq S_L(x)$;*

(b) *$X_I \subseteq X_G$ and $X_I \subseteq X_L$;*

(c) *$S_I(0) \subseteq X_I$ and $S_G(0) \subseteq X_G$.*

Proof. (a) Follows from the increasingly restrictive definitions of $S_L(x)$, $S_G(x)$, and $S_I(x)$, $S_L(x)$.

(b) Follows from (a) and the property P1(a) that X_I, X_G, and X_L are the subclosures of $S_I(0)$, $S_G(0)$, and $S_L(0)$.

(c) Follows from the fact that $S_I(0)$ and $S_G(0)$ (but not $S_L(0)$) are subsets of Z^n_+.

Our valid inequalities will be constructed from functions π defined on X such that $X_I \subseteq X$ and $X \subseteq Z^n$. For convenience, denote by δ^j the vector

$$\delta^j_i = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}, \quad i = 1, \ldots, n.$$

Theorem 1. *Let π be a subadditive function on X_I, that is,*

$$\pi(x^1 + x^2) \leq \pi(x^1) + \pi(x^2), \tag{5}$$

for all $x^1, x^2 \in X_I$ such that $(x^1 + x^2) \in X_I$. Then the inequality

$$\sum_{j=1}^{n} \pi_j x_j \geq \pi_0 \tag{6}$$

is valid, where

$$\pi_j = \pi(\delta^j), \tag{7}$$

$$\pi_0 \leq \min\{\pi(x) \mid x \in S_I(0)\}. \tag{8}$$

Note 1. If $\delta^j \notin X_I$ or, equivalently, $S_I(\delta^j) = \emptyset$, then π_j can be set arbitrarily small $(-\infty)$. In this case, x_j can be eliminated from the problem by setting it to zero.

Note 2. If $0 \in X_I$, $\pi(0) = 0$ is required.

Proof. It suffices to show that

$$\pi(x) \leq \sum_{j=1}^{n} \pi_j x_j, \quad x \in X_I, \tag{9}$$

since if $y \in S_i(0)$, then $y \in X_I$ by property P2(e) and by (9)

$$\sum_{i=1}^{n} \pi_i y_i \geq \pi(y) \geq \pi_0$$

where the second inequality is by (8). Hence, we prove (9) for all $x \in X_I$.

Using δ^i gives

$$x = (x_1, \ldots, x_n) = x_1\delta^1 + \cdots + x_n\delta^n.$$

By $x \in X_I$ and subinclusion, if $x_i > 0$, then $\delta^i \in X_I$ and so does every

$$x' = x'_1\delta^1 + \cdots + x'_n\delta^n$$

for $0 \leq x'_i \leq x_i$, x'_i integer.

The proof can be done by induction on

$$T = \sum_{i=1}^{n} x_i.$$

If $0 \in X_I$, then for $x = 0$, (9) reduces to $\pi(0) \leq 0$. By subadditivity, $\pi(0) \geq 0$, so we must require $\pi(0) = 0$ in this case (see note 2).

For $T = 1$, (9) follows from $\pi_i = \pi(\delta^i)$ whenever $\delta^i \in X_I$.

The induction step is exactly as appears in the proofs of theorem 1.5 of [8] or theorem 2.2 of [5].

Remark 1. The strongest inequality is given by taking π_0 equal to the minimum value of $\sum \pi_i x_i$ in (6) over all integer programming feasible solutions x. However, finding that minimum is as hard, in general, as the original ILP. A possible weaker inequality is given by taking π_0 to be the minimum given in (8). Finding that minimum is also a constrained minimization problem of the same order of difficulty as the original ILP. In practice we use, for convenience, a superset $S \supseteq S(0)$ to yield a weaker, but valid, inequality.

Remark 2. The direct application of Theorem 1 to construct valid inequalities can become cumbersome (even when π is known). The formal expressions used to define π can be complicated and their evaluations difficult. In [4], comparisons of cutting planes of this form are given. Here, the particular π used is motivated by a desire to be able to evaluate $\pi(x)$ easily.

3. Generalized gauge functions

A function which is crucial to our development here is the generalized diamond gauge function. It allows considerable flexibility to the subadditive functions Δ and π to be constructed from it while still being easy to evaluate. We first define this function and give some of its properties.

Definition 5. Given $2m_1 + 2m_2$ real numbers

$$\gamma_1^+, \ldots, \gamma_{m_1}^+, \gamma_1^-, \ldots, \gamma_{m_1}^-, \alpha_1^+, \ldots, \alpha_{m_2}^+, \alpha_1^-, \ldots, \alpha_{m_2}^-$$

define the *generalized diamond gauge function* D *on* $\mathbb{R}^n$ *to* $\mathbb{R}$ *by*

$$D(x) = \max_{\gamma, \alpha} \{\gamma G x + \alpha A x\} \tag{10}$$

where the maximum is taken over all $2^{m_1 + m_2}$ possible values:

$$\gamma_i = \gamma_i^+ \text{ or } -\gamma_i^-,$$

$$\alpha_i = \alpha_i^+ \text{ or } -\alpha_i^-.$$

We require that

$$\gamma_i^+ \geq 0 \text{ and } \gamma_i^- \geq 0, \text{ and} \tag{11}$$

$$\alpha_i^+ + \alpha_i^- \geq 0. \tag{12}$$

For our example,

$$D(x) = \max \begin{Bmatrix} \frac{1}{2}\gamma^+ x_1 + \frac{1}{2}\gamma^+ x_2 \\ -\frac{1}{2}\gamma^- x_1 - \frac{1}{2}\gamma^- x_2 \end{Bmatrix} + \max \begin{Bmatrix} -4\alpha^+ x_1 - 11\alpha^+ x_2 \\ +4\alpha^- x_1 + 11\alpha^- x_2 \end{Bmatrix}$$

$$= \max \begin{cases} \gamma^+ u + \alpha^+ \xi \\ \gamma^+ u - \alpha^- \xi \\ -\gamma^- u + \alpha^+ \xi \\ -\gamma^- u - \alpha^- \xi \end{cases}$$

where $u = \frac{1}{2}x_1 + \frac{1}{2}x_2$ and $\xi = -4x_1 - 11x_2$. The $\gamma^+, \gamma^-, \alpha^+, \alpha^-$ are parameters of the function D.

Properties of D

Property P3. *The generalized diamond gauge* D *contains* $2m_1 + 2m_2$ *parameters* $\gamma^+, \gamma^-, \alpha^+, \alpha^-$ *satisfying* $2m_1 + m_2$ *inequalities* (11) *and* (12).

Property P4. D *is piecewise linear and continuous.*

Property P5. D *is convex and positively homogeneous.*

Proof. Both convexity and positive homogeneity follow from the definition of D as the maximum of the $2^{m_1+m_2}$ linear functions $(\gamma G + \alpha A)x$, each of which goes through the origin. By positively homogeneous is meant $D(\lambda x) = \lambda D(x)$ for $\lambda \geq 0$.

A function f which is non-negative, convex, and positively homogeneous is called a *gauge* by Rockafellar [12]. Dropping non-negativity, we call f a *generalized* gauge. A generalized gauge can be characterized by being the support function (see [14], section 13, particularly theorem 13.2) of a non-empty convex set (see [11] also).

Before continuing with D, we digress to give one result which is true of any generalized gauge f (see [12], theorem 4.7). We include its proof here for completeness.

Property P6. *A generalized gauge f is subadditive.*

Proof. We need to show

$$f(x+y) \leq f(x) + f(y), \quad x, y \in \mathbb{R}^n,$$

whenever f is convex and positively homogeneous. By convexity,

$$f(x+y) = f(\tfrac{1}{2}2x + \tfrac{1}{2}2y) \leq \tfrac{1}{2}f(2x) + \tfrac{1}{2}f(2y).$$

By positive homogeneity,

$$\tfrac{1}{2}f(2x) + \tfrac{1}{2}f(2y) = f(x) + f(y),$$

completing the proof.

Property P7. *Given $x \in \mathbb{R}^n$, let $u = Gx$ and $\xi = Ax$. Then*

$$D(x) = \sum_{i=1}^{m_1} \max\{\gamma_i^+ u_i, -\gamma_i^- u_i\} + \sum_{i=1}^{m_2} \max\{\alpha_i^+ \xi_i, -\alpha_i^- \xi_i\}. \tag{13}$$

Proof. Using (10),

$$\begin{aligned} D(x) &= \max_{\gamma, \alpha} \{\gamma G x + \alpha A x\} \\ &= \max_{\gamma, \alpha} \left\{ \sum_{i=1}^{m_1} \gamma_i u_i + \sum_{i=1}^{m_2} \alpha_i \xi_i \right\} \\ &= \sum_{i=1}^{m_1} \max\{\gamma_i u_i \mid \gamma_i = \gamma_i^+ \text{ or } -\gamma_i^-\} \\ &\quad + \sum_{i=1}^{m_2} \max\{\alpha_i \xi_i \mid \alpha_i = \alpha_i^+ \text{ or } -\alpha_i^-\} \end{aligned}$$

since the maximization can be done separately for each i.

Property P8. *Given $x \in \mathbb{R}^n$, let $u = Gx$ and $\xi = Ax$. Then $D(x) = \gamma G x + \alpha A x$ for γ and α given by*

$$\gamma_i = \sigma(u_i, \gamma_i^+, \gamma_i^-) \text{ and } \alpha_i = \sigma(\xi_i, \alpha_i^+, \alpha_i^-) \tag{14}$$

where σ is the sign transfer function, with arguments q, w^+, w^-, defined by:

$$\sigma(q, \sigma^+, \sigma^-) = \begin{cases} \sigma^+, & q > 0, \\ 0, & q = 0, \\ -\sigma^-, & q < 0. \end{cases}$$

This sign transfer function is related to the fortran function SIGN by

$$\mathrm{SIGN}(x1, x2) = \sigma(x2, x1, -x1)$$

and to the usual absolute value function by

$$|x| = \sigma(x, x, -x).$$

We remark that the function of one variable

$$f(q) = q\sigma(q, \sigma^+, \sigma^-)$$

is convex if and only if $\sigma^+ + \sigma^- \geq 0$.

Proof of P8. The proof is from P7 and substituting

$$\max\{\gamma_i^+ \omega_i, -\gamma_i^- \omega_i\} = \omega_i\sigma(\omega_i, \gamma_i^+, \gamma_i^-), \text{ and}$$

$$\max\{\alpha_i^+ \xi_i, -\alpha_i^- \xi_i\} = \xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-),$$

which follows from $\alpha_i^+ + \alpha_i^- \geq 0$ and $\gamma_i^+ + \gamma_i^- \geq 0$.

We remark that only $\gamma_i^+ + \gamma_i^- \geq 0$ is needed, rather than the stronger (11). We require (11) because it will eventually be needed for other reasons.

The main reason for using this D is property P7 (or P8) which allows an overall maximum to be taken coordinate-wise. Thus a maximum over $2^{m_1+m_2}$ linear functions can be effected by taking $m_1 + m_2$ pairwise maxima. For further generalizations of diamond gauges [3] see [4].

We now turn to a family of subadditive functions which can be defined from the generalized diamond gauge. The prototype here is the Gomory mixed integer cut for the case where $m_1 = 1$ and $m_2 = 0$. To derive that cut, let the diamond gauge have $\gamma^+ = 1/g^0$ and $\gamma^- = 1/(1 - g^0)$. In this case, $\Delta_P(x)$ or $\Delta_G(x)$, to be defined below, only depend on the single parameter $u = Gx - \sum g^j x_j$ and are given by

$$\Delta(x) = \begin{cases} \dfrac{F(u)}{g^0}, & 0 \leq F(u) \leq g^0, \\[2ex] \dfrac{1 - F(u)}{1 - g^0}, & g^0 < F(u) < 1, \end{cases}$$

where $F(u)$ is the fractional part of u.

Definition 6. For the generalized diamond gauge D, define the *subadditive diamond functions* $\Delta_{\geq}, \Delta_{\leq}, \Delta_P$ and Δ_G by

$$\Delta_0(x) = \min_z \{D(z) \mid z \text{ satisfies: } z \geq 0,\ z \text{ integer},\ Gz \equiv Gx \pmod 1,\ Az \geq Ax\}, \tag{15}$$

$$\begin{aligned}\Delta_1(x) = \min_z \{\gamma u \mid u = Gz \text{ for } z \geq 0, \text{ integer, and } Gz \equiv Gx \pmod 1\}\\ + \min_z \{\alpha\xi \mid \xi = Az \text{ for } z \geq 0, \text{ integer, and } Az \geq Ax\},\end{aligned} \tag{16}$$

$$\Delta_2(x) = \min_u \{\gamma u \mid u \equiv Gx\} + \min_\xi \{\alpha\xi \mid \xi = Az \text{ for } z \geq 0 \text{ and } Az \geq Ax\}, \tag{17}$$

$$\Delta_3(x) = \min_u \{\gamma u \mid u \equiv Gx\} + \min_\xi \{\alpha\xi \mid \xi \geq Ax\}, \tag{18}$$

where γ and α in (16), (17), and (18) are given by (14) as functions of u and ξ.

Properties of Δ

Property P9. $\Delta_0(x) \geq \Delta_1(x) \geq \Delta_2(x) \geq \Delta_3(x)$.

Proof. $\Delta_1(x)$ is obtained from $\Delta_0(x)$ by splitting the minimization of the sum $\gamma Gy + \alpha Ay$ into the sum of two minimization with each minimization taken over a larger set of y's. Hence, $\Delta_0(x) \geq \Delta_1(x)$.

$\Delta_2(x)$ is obtained from $\Delta_1(x)$ by weakening the constraints of each minimization.

$\Delta_3(x)$ is obtained from $\Delta_2(x)$ by further relaxing the second constraint set.

Property P10. $\Delta_2(x) = \sum_{i=1}^{m} \min\{\gamma_i^+ F(u_i), -\gamma_i^-(F(u_i) - 1)\}$

$$+ \min\left\{\sum_{i=1}^{m} \xi_i \sigma(\xi, \alpha_i^+, \alpha_i^-) \,\middle|\, \xi = Ay,\ y \geq 0,\ Ay \geq Ax\right\},$$

where $u = Gx$ *and* $F(u_i)$ *is the fractional part of* u_i: $F(u_i) \equiv u_i \pmod 1$, *and* $0 \leq F(u_i) < 1$.

Proof. The result

$$\min\{\gamma f \mid f \equiv Gx\} = \sum_{i=1}^{m} \min\{\gamma_i^+ F(u_i), -\gamma_i^-(F(u_i) - 1)\}$$

follows from $\gamma_i^+ \geq 0$, $\gamma_i^- \geq 0$; that is, from (11). Secondly,

$$\min_y \{\alpha Ay \mid y \geq 0, Ay \geq Ax\} = \min\left\{\sum_{i=1}^{m} \xi_i \sigma(\xi, \alpha_i^+, \alpha_i^-) \,\middle|\, \xi = Ay, y \geq 0, Ay \geq Ax\right\},$$

follows from property P8 and (14). This latter form of the minimization should make it clear that in ξ and y the problem is a linear program with a separable, piece-wise linear, convex objective function. Convexity follows from convexity of $\xi\sigma(\xi, \alpha^+, \alpha^-)$.

Property P11. *For $\gamma_i^+ \geq 0$ and $\gamma_i^- \geq 0$, define*

$$R_i = \begin{cases} \dfrac{\gamma_i^-}{\gamma_i^+ + \gamma_i^-}, & \text{when } \gamma_i^+ + \gamma_i^- > 0,\\[2ex] 1, & \text{when } \gamma_i^+ + \gamma_i^- = 0. \end{cases}$$

Then for $x \in \mathbf{R}^n$ and $u = Gx$,

$$\Delta_1(x) = \sum_{i=1}^{m} \sigma(R_i - F(u_i), \gamma_i^+ F(u_i), \gamma_i^-(F(u_i) - 1))$$

$$+ \min\left\{ \sum_{j=1}^{n} \xi_j \sigma(\xi_j, \alpha_j^+, \alpha_j^-) \,\middle|\, \xi = Ay, y \geq 0, Ay \equiv Ax \right\}.$$

Proof. This property is proven by showing

$$\min\{\gamma_i^+ F(u_i), -\gamma_i^-(F(u_i) - 1)\} = \sigma(F(u_i) - R_i, \gamma_i^+ F(u_i), \gamma_i^-(F(u_i) - 1)).$$

If $\gamma_i^+ = \gamma_i^- = 0$, then equality holds trivially. Hence, suppose $\gamma_i^+ + \gamma_i^- > 0$. Then the condition

$$\gamma_i^+ F(u_i) \geq -\gamma_i^-(F(u_i) - 1)$$

holds if, and only if,

$$F(u_i)(\gamma_i^+ + \gamma_i^-) \geq \gamma_i^-, \text{ or}$$

$$F(u_i) \geq R_i \text{ by } \gamma_i^+ + \gamma_i^- > 0, \text{ or}$$

$$F(u_i) - R_i \geq 0.$$

Hence, we can conclude: if, and only if,

$$\sigma(F(u_i) - R_i, \gamma_i^+ F(u_i), \gamma_i^-(F(u_i) - 1)) = \gamma_i^+ F(u_i).$$

Property P12. $\Delta_1(x) = \displaystyle\sum_{i=1}^{m} \sigma(R_i - F(u_i), \gamma_i^+ F(u_i), \gamma_i^-(F(u_i) - 1))$

$$+ \sum_{j=1}^{n} b_j \sigma(b_j, \alpha_j^+, \min\{0, \alpha_j^-\})$$

where R_i and u are as in Property P11 and where $b = Ax$, provided

$$\alpha_j^+ \geq 0.$$

Proof. The first half of the expression for $\Delta_1(x)$ is proven exactly as in P11. The second half follows from

$$\min\left\{\sum_{i=1}^{m_2} \xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-)\,\Big|\, \xi \geq Ax\right\} = \sum_{i=1}^{m_2} \min\{\xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-)\,|\, \xi_i \geq (Ax)_i\}.$$

If $b = Ax$ has $b_i \geq 0$, then by $\alpha_i^+ \geq 0$,

$$\xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-) \geq \alpha_i^+ b_i.$$

If $b_i < 0$, then

$$\min\{\xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-)\,|\, \xi_i \geq b_i\} = \begin{cases} 0, & \text{if } \alpha_i^- > 0,\\ b_i\xi_i, & \text{if } \alpha_i^- \leq 0.\end{cases}$$

Hence, in either case

$$\min\{\xi_i\sigma(\xi_i, \alpha_i^+, \alpha_i^-)\,|\, \xi_i \geq b_i\} = b_i\sigma(b_i, \alpha_i^+, \min\{0, \alpha_i^-\}).$$

In order to use Δ_0 or Δ_1, only $\gamma_i^+ + \gamma_i^- \geq 0$ and $\alpha_i^+ + \alpha_i^- \geq 0$ need be imposed; Δ_2 requires $\gamma_i^+ \geq 0$, $\gamma_i^- \geq 0$, and $\alpha_i^+ + \alpha_i^- \geq 0$; and Δ_3 requires even further that $\alpha_i^- \geq 0$.

In practice, either Δ_2 or Δ_3 is used because their evaluations are not difficult. The strongest function for our purposes would be Δ_0, but the evaluations $\Delta_0(x)$ are, in general, as difficult as the original ILP.

The next property is true for all four Δ's, but will only be proven for Δ_2.

Property P13. *Δ_2 is subadditive.*

Proof. We refer back to the definition (17). We need to show

$$\Delta_2(x^1) + \Delta_2(x^2) \geq \Delta_2(x^1 + x^2) = \Delta_2(x^3),$$

where $x^3 = x^1 + x^2$. But for some $y^1, y^2, y^3 \geq 0$ with

$$Ay^1 \geq Ax^1,\ Ay^2 \geq Ax^2,\ Ay^3 \geq Ax^3,$$

and $f^1 \geq Gx^1$, $f^2 \geq Gx^2$, $f^3 \geq Gx^3$,

$$\Delta_2(x^i) = \gamma f^i + \alpha Ay^i, \quad i = 1, 2, 3.$$

The minima in (17) are achieved because $\gamma_i^+ \geq 0$, $\gamma_i^- \geq 0$ and $\alpha_i^+ + \alpha_i^- \geq 0$ (see property P11). Hence,

$$\begin{aligned}\Delta_2(x^1) + \Delta_2(x^2) &= \gamma f^1 + \alpha Ay^1 + \gamma f^2 + \alpha Ay^2\\ &= \gamma(f^1 + f^2) + \alpha A(y^1 + y^2)\\ &\geq \gamma f^3 + \alpha Ay^3 = \Delta_2(x^3),\end{aligned}$$

because

$$G(f^1 + f^2) \geq Gx^1 + Gx^2 = Gx^3,$$

$$A(y^1 + y^2) \geq Ax^1 + Ax^2 = Ax^3.$$

The proof that Δ_2 is subadditive is even simpler. To prove Δ_0 and Δ_1 are subadditive, one needs to observe that when an infinite number of y's satisfy the constraints, the minima in (15) and (16) need not be achieved. However, for a given x, a y can be found so that, for example, $D(y) - \varepsilon = \Delta_0(x)$ for any preassigned $\varepsilon > 0$. Then, the proof is much as before.

4. The generator set

In the preceding sections we have formulated a functional framework to be used for the construction of valid inequalities which take into account the integrality requirements (i.e. the group structure) and all LP constraints of the ILP. For practical reasons we have focused our attention on a particular class of subadditive functions called *subadditive diamond functions* built from generalized diamond gauges D. However, D is convex, a property which is not required by the subadditive theory for valid inequalities. In the present section, we use an ad hoc device (viz. the *generator set* X_G) to produce non-convex subadditive functions π: as for the method [5] which solves the group problem, these π functions generate enumerative inequalities and combine the concepts of group structure, cutting plane and enumeration.

Throughout this section we need only assume Δ to be subadditive, but it will become clear in Section 6 that the following developments would be meaningless (certainly of no practical value) if a concrete example (i.e. diamond gauges) were not available with specifically useful additional properties.

We now introduce the *generator set* X_G, which is a finite set of non-negative integer vectors $y \in \mathbf{Z}^n_+$. Initially, X_G will only be required to be subinclusive; that is, if $y \in X_G$, $0 \leqslant y' \leqslant y$, and y' integer, then $y' \in X_G$. Subsequently, X_G will be constructed sequentially as needed by the algorithm.

Definition 7. Define $\pi(x)$, $x \in \mathbf{R}^n$, from a subadditive Δ on $\mathbf{R}^n$, a finite generator set X_G, and an arbitrary function d on X_G by

$$\pi(x) = \min_{y \in I(x)} \{d(y) + \Delta(x - y)\} \tag{19}$$

where $I(x) \subseteq X_G$ for all x.

Two particularly useful ways of defining $I(x)$ will be used: $I(x) = X_G$ for all x and $I(x) = X_G \cap \underline{S}(x)$, where $\underline{S}(x)$ is the subclosure of x:

$$\underline{S}(x) = \{y \text{ integer} : 0 \leqslant y \leqslant x\}.$$

Whereas π is defined for all $x \in \mathbf{R}^n$, X_G and, hence, $I(x)$ are always subsets of the integer points $\mathbf{Z}^n_+ \subset \mathbf{R}^n$. When $I(x) = X_G \cap \underline{S}(x)$, then clearly $\pi(x) = +\infty$ for all $x \notin \mathbf{R}^n_+$, so π may as well be considered to be only defined on $\mathbf{R}^n_+$.

Theorem 2. *If $I(x)=X_L$ for all x, and if*

$$\pi(y^1+y^2)\le d(y^1)+d(y^2)\quad \textit{for all } y^1, y^2\in X_L,$$

then π is subadditive.

The proof is virtually the same as that of [5, Theorem 3.11] and is similar to the proof of Theorem 3 to follow. For those reasons, it is not given here.

Before giving Theorem 3, a lemma is needed.

Lemma 1. *If $I(x^1)\subseteq I(x^2)$, then $\pi(x^1)+\Delta(x^2-x^1)\ge\pi(x^2)$.*

Proof. By (19) and finiteness of X_L, and hence of $I(x)\subseteq X_L$, there is some $y^1\in I(x^1)$ such that

$$\pi(x^1)=d(y^1)+\Delta(x^1-y^1).$$

Hence,

$$\begin{aligned}\pi(x^1)+\Delta(x^2-x^1)&=d(y^1)+\Delta(x^1-y^1)+\Delta(x^2-x^1)\\&\ge d(y^1)+\Delta(x^2-y^1).\end{aligned}$$

Now, by $y^1\in I(x^1)\subseteq I(x^2)$, $y^1\in I(x^2)$ and

$$\pi(x^2)\le d(y^1)+\Delta(x^2-y^1)$$

from (19). Therefore,

$$\pi(x^1)+\Delta(x^2-x^1)\ge\pi(x^2).$$

Theorem 3. *If $I(x)=X_L\cap S(x)$, all $x\in\mathbf{R}^m$, and if*

$$\pi(y^1+y^2)\le d(y^1)+d(y^2),\quad \textit{all } y^1, y^2\in X_L, \tag{20}$$

then π is subadditive.

Proof. For $x^1, x^2\in\mathbf{R}^m$, we wish to show that

$$\pi(x^1+x^2)\le\pi(x^1)+\pi(x^2).$$

Let $y^1\in I(x^1)$ and $y^2\in I(x^2)$ give the minimum in (19) defining $\pi(x^1)$ and $\pi(x^2)$. Then

$$\begin{aligned}\pi(x^1)+\pi(x^2)&=d(y^1)+d(y^2)+\Delta(x^1-y^1)+\Delta(x^2-y^2)\\&\ge\pi(y^1+y^2)+\Delta(x^1+x^2-(y^1+y^2))\end{aligned}$$

using (20) and subadditivity of Δ.

Now, $y^1\in I(x^1)$, and thus $y^1\le x^1$. Similarly, $y^2\le x^2$. Hence, $y^1+y^2\le x^1+x^2$ and therefore

$$I(y^1 + y^2) \subseteq I(x^1 + x^2).$$

Applying Lemma 1 gives

$$\pi(y^1 + y^2) + \Delta(x^1 + x^2 - (y^1 + y^2)) \ge \pi(x^1 + x^2).$$

Therefore,

$$\pi(x^1) + \pi(x^2) \ge \pi(x^1 + x^2),$$

completing the proof.

In [5], we showed that the condition (20) could be relaxed to a small subset of $y^1, y^2 \in X_E$. Theorem 4.3 there can be extended to this problem for the case $I(x) = X_E$. Here, we give the development for $I(x) = X_E \cap \underline{S}(x)$. The resulting Theorem 4 below holds true in either case.

Henceforth, we specialize $d(y)$ to be

$$d(y) = \sum_{j=1}^{n} c_j y_j \qquad (21)$$

where c_j is the cost coefficient of x_j in (1).

Lemma 2. *For X_E subinclusive, $I(x) = X_E \cap \underline{S}(x)$, and d given by (21), the set*

$$X_E^+ = \{y \in X_E \mid d(z) + \Delta(y - z) \ge d(y), \text{ all } 0 \le z \le y\}$$

is also subinclusive.

Proof. Let $y \in X_E^+$. Then,

$$d(z) + \Delta(y - z) \ge d(y)$$

for all $0 \le z \le y$. Hence,

$$\Delta(y - z) \ge d(y) - d(z) = d(y - z)$$

since d is linear. In other words,

$$\Delta(z) \ge d(z), \text{ all } 0 \le z \le y. \qquad (22)$$

In order to prove that X_E^+ is subinclusive, we need to show that for all $z \le y$, we have

$$d(z') + \Delta(z - z') \ge d(z), \ 0 \le z' \le z.$$

Using the characterization (22) of X_E^+, we need to show

$$\Delta(z'') \ge d(z''), \text{ all } 0 \le z'' \le z.$$

However, condition (22) for y applies to z'' since these z'' are less than y as well as less than z. The proof is completed.

Given a subinclusive set X_E, define the *candidate set* X_C by

$$X_F = \{x \in \mathbf{Z}^n_+ : x \notin X_E \text{ and } S(x) - \{x\} \subseteq X_E\},$$

that is, if $x \in X_F$ and $y < x$, then $y \in X_E$, assuming $x, y \in \mathbf{Z}^n_+$. The $x \in X_F$ will clearly be pairwise incomparable.

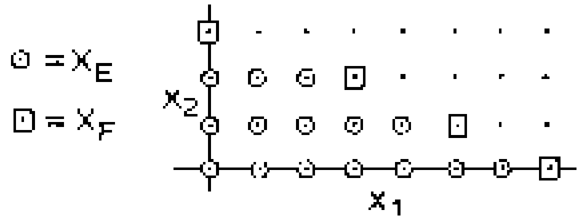


Fig. 1

Conversely, given any set X_F of incomparable elements, the generator set X_E whose candidate set would be X_F can be characterized as follows: $y \in X_E$ if and only if for every $x \in X_F$ either $x > y$ or x and y are incomparable (see Fig. 1).

Lemma 3. *For a subinclusive X_E and its candidate set X_F, let*

$$Z_+(X_F) = \left\{ z \;\middle|\; z = \sum k_i x^i, \text{ where } x^i \in X_F \text{ and } k_i \geq 0, \text{ integer} \right\}.$$

Then

$$\mathbf{Z}^n_+ = Z_+(X_F) + X_E.$$

Proof. The lemma says that for any $z \in \mathbf{Z}^n_+$ there is some $x \in \mathbf{Z}_+(X_F)$ and $y \in X_E$ such that $z = x + y$.

Consider z and let $z^0 \leq z$ be any maximal element in $Z_+(X_F) \cap S(z)$; that is, $z^0 \leq z$, $z^0 \in \mathbf{Z}_+(X_F)$. Then, for any $x \in X_F$, $z^0 + x$ is either greater than z or incomparable to z since $z^0 + x \in \mathbf{Z}_+(X_F)$. Therefore, for any $x \in X_F$, x is either greater than $z - z^0$ or incomparable to $z - z^0$. Hence, $z - z^0 \in X_E$, using the characterization of X_E given X_F.

Theorem 4. *If $I(x) = X_E \cap S(x)$, if $d(y) = cy$, and if*

$$\pi(y^1 + y^2) \leq d(y^1) + d(y^2), \quad y^1, y^2 \in X_E, \quad y^1 + y^2 \in X_F, \tag{23}$$

then π is subadditive.

Proof. Here, X_E is given from X_F as in Lemma 2. By that lemma, X_E is subinclusive. The X_F is the candidate set for X_E.

By (23) and by linearity of d, for $x \in X_F$,

$$\pi(x) \leq d(x).$$

By definition (19) of π, for some $y \leq x$,

$$d(y) + \Delta(x-y) \le d(x),$$

or

$$\Delta(x-y) \le d(x-y).$$

By the characterization (22) of X_E^*, $\Delta(x-y) > d(x-y)$ whenever $0 \ne x-y \le x$. Hence, the only possible y is $y = 0$, and, therefore, $\Delta(x) \le d(x)$ follows whenever $x \in X_E^*$.

Consider now $Z_+(X_E^*)$, as in Lemma 3. If x^1 and $x^2 \in X_E^*$, then

$$\Delta(x^1 + x^2) \le \Delta(x^1) + \Delta(x^2) \le d(x^1) + d(x^2) = d(x^1 + x^2).$$

Continuing by induction, one has

$$\Delta(x) \le d(x),$$

all $x \in Z_+(X_E^*)$.

By Lemma 3, for any $z \in Z_+^n$,

$$z = x + y$$

for some $x \in Z_+(X_E^*)$ and $y \in X_E^*$. By (19),

$$\begin{aligned} \pi(z) &\le d(y) + \Delta(x) \\ &\le d(y) + d(x), \quad \text{by } x \in Z_+(X_E^*) \\ &= d(y+x) = d(z). \end{aligned}$$

Consider any $y^1, y^2 \in X_E^*$. Then for $z = y^1 + y^2$, $\pi(y^1 + y^2) \le d(y^1) + d(y^2)$. By Theorem 3, π is subadditive.

Corollary 1. *Given a subadditive Δ on $\mathbf{R}_+^n$ and a linear d on $\mathbf{R}_+^n$, let*

$$X_E = \{y \in \mathbf{Z}_+^n \mid \Delta(z) > d(z) \text{ all } 0 < z \le y\}.$$

Assume X_E is finite. Then π given by (19), *using $I(x) = X_E \cap S(x)$, is subadditive.*

Proof. The X_E given here is subinclusive by Lemma 2 and condition (22) for X_E^* there. For this X_E, $X_E^* = X_E$. Hence, we need only show

$$\Delta(x) \le d(x), \; x \in X_E.$$

For $x \in X_E$ and $y \le x$, $y \in X_E$ so

$$\Delta(y) > d(y), \text{ if } y \ne 0.$$

Hence, unless $\Delta(x) \le d(x)$, this x would also be in X_E.

5. Bounds

Theorem 4, combined with Theorem 1, says that if

$$\Delta(x) \leq d(x)$$

for every $x \in X_D^c$, then

$$\sum_{j=1}^{n} \pi_j x_j \geq \pi_0$$

is a valid inequality whenever (8) holds; that is,

$$\pi_0 \leq \min\{\pi(x) \mid x \in S_1(0)\}.$$

This π_0 will be a lower bound on the optimum objective function value. To prove this result, recall that $\pi_j = \pi(\delta^j)$. Hence, the inequality $\pi_j \leq c_j$ will hold because either

(i) $\delta^j \in X_D$ and then $\pi(\delta^j) = d(\delta^j) = c_j$, or

(ii) $\delta^j \notin X_D$ and then $\delta^j \in X_D^c$ so $\pi(\delta^j) = \Delta(\delta^j) \leq d(\delta^j) = c_j$.

By $\pi_j \leq c_j$, π_0 is a lower bound on the optimum objective value z in (1) because for every $x \in S_1(0)$

$$\pi_0 \leq \sum_{j=1}^{n} \pi_j x_j \leq \sum_{j=1}^{n} c_j x_j \text{ by } \pi_j \leq c_j.$$

We now consider the computations required by (8) for π constructed from Δ_0, Δ_1, Δ_2, and Δ_3.

5.1 For Δ_0

$$\pi_0 = \min_{x \in S_1(0)} \left\{ \min_{y \in X_D} \{d(y) + \Delta_0(x - y)\} \right\}$$

$$= \min_{y \in X_D} \left\{ d(y) + \min_{x \in S_1(0)} \{\min\{D(z) \mid z \geq 0, \text{ integer}, Gz = G(x - y), \text{ and } Az \geq A(x - y)\}\} \right\}.$$

Substituting z in place of $z + y$ gives

$$\pi_0 = \min_{y \in X_D} \left\{ d(y) + \min_{x \in S_1(0)} \{\min\{D(z - y) \mid z \geq y, \text{ integer}, Gz = Gx, \text{ and } Az \geq Ax\}\} \right\},$$

or, finally,

$$\pi_0 = \min_{y \in X_D} \left\{ d(y) + \min_{z \in S_1(0)} \{D(z - y)\} \right\}. \tag{24}$$

To show that (24) is equivalent to the expression just above it requires showing that the constraints on z:

$$z \geq y, \quad \text{integer}, \quad Gz = Gx, \quad Az \geq Ax,$$

for $x \in S_I(0)$, are equivalent to the seemingly weaker restrictions:

$$z \geq y, \quad \text{integer}, \quad Gz = g^0, \quad Az \geq a^0.$$

In order to prove equivalence of these two sets of constraints, we must show that for a z satisfying the latter set, there is some $x \in S_I(0)$ for which z satisfies the former set. However, this x can be taken to be z since $z \in S_I(y)$ implies that $z \in S_I(0)$.

We consider the bounds given for our example with each of Δ_0, Δ_1, and Δ_2 in turn, using $X_F = \{(0,0)\}$ and $X_F = \{(1,0),(0,1)\}$.

By (24) and $X_F = \{(0,0)\}$,

$$\pi = d(0) - \min_{z \in S_I(0)} \{D(z)\}$$
$$= -D(3,0),$$

since $S_I(0) = \{(3,0)\}$. Now, for $g = \frac{3}{4}x_1 + \frac{1}{4}x_2 = \frac{3}{4}(3) + \frac{1}{4}(0) = 2\frac{1}{4}$ and $a = -4x_1 - 11x_2 = -4(3) - 11(0) = -12$,

$$D(3,0) = \max \begin{cases} -\gamma^-(2\frac{1}{4}) + \alpha^+(-12), \\ \gamma^+(2\frac{1}{4}) - \alpha^-(-12), \\ -\gamma^-(2\frac{1}{4}) - \alpha^-(-12), \\ -\gamma^+(2\frac{1}{4}) - \alpha^+(-12), \end{cases}$$

from Section 3. The constraints $\Delta_0(x) \leq d(x)$, $x \in X_F$, are satisfied provided

$$\max\begin{Bmatrix} \gamma^+ \frac{3}{4} \\ \gamma^- \frac{1}{4} \end{Bmatrix} + \max\begin{Bmatrix} \alpha^+ & 0 \\ \alpha^- & 4 \end{Bmatrix} \leq 1,$$

$$\max\begin{Bmatrix} \gamma^+ \frac{1}{4} \\ \gamma^- \frac{3}{4} \end{Bmatrix} + \max\begin{Bmatrix} \alpha^+ & 0 \\ \alpha^- & 11 \end{Bmatrix} \leq 1$$

for $\alpha^\pm \geq 0$. For example, $\gamma^+ = \frac{4}{3}$, $\gamma^- = \frac{4}{3}$, $\alpha^+ = \alpha^- = 0$. Then, $\pi_0 = 3$, which is the optimum objective value of the example.

Evaluating π_0 by (24) is as hard, in general, as solving the original ILP. Weaker versions of it will be developed from Δ_1, Δ_2, and Δ_3.

5.2 For Δ_1

$$\pi_1 = \max_{y \in X_F} \Big\{ d(y) + \min_{x \in S_I(0)} [\min\{\gamma Gz \mid z \geq 0, \text{ integer}, Gz = G(x-y)\}$$
$$- \min\{\alpha A z \mid z \geq 0, z \text{ integer}, Az \geq A(x-y)\}] \Big\}$$

Substituting z in place of $z + y$ and simplifying gives

$$\pi_0 = \min_{y \in S_0} \left\{ d(y) + \min\left\{ \gamma G(z-y) \mid z \ge y,\ \text{integer},\ Gz = g^0 \right. \right.$$
$$\left. \left. + \min_{x \in S_I(0)} \{\min\{\alpha A(z-y) \mid z \ge y,\ z \text{ integer},\ Az \ge Ax\}\right\} \right\}.$$

If $x \in S_I(0)$ is weakened to $x \in S$ where

$$S = \{x \text{ integer} \mid x \ge 0,\ Ax \ge a^0\}$$

then the expression simplifies to

$$\pi_0 = \min_{y \in S_0} \{d(y) + \min\{\gamma G(z-y) \mid z \in S_G(y)\}$$
$$+ \min\{\alpha A(z-y) \mid z \ge y,\ z \text{ integer},\ Az \ge a^0\}\}. \tag{25}$$

Using the same values $\gamma^+ = \frac{1}{2}$, $\gamma^- = \frac{1}{2}$, $\alpha^- = \alpha^+ = 0$, (25) gives

$$\pi_0 = \min\left\{ \begin{matrix} \gamma^+ u, & u \ge 0 \\ -\gamma^- u, & u < 0 \end{matrix} \;\middle|\; u = \tfrac{1}{2}z_1 + \tfrac{1}{2}z_2,\ z \in S_G(0) \right\}$$
$$= \min\{\gamma^+ \tfrac{1}{2}, \gamma^- \tfrac{1}{2}\} = \tfrac{1}{4}.$$

This bound, not unexpectedly, is not as large as the bound from Δ_2.

5.3

Turning to Δ_3, we obtain

$$\pi_0 = \min_{y \in S_0} \left\{ d(y) + \min_{x \in S_I(0)} \{\min\{\gamma f \mid f = G(x-y)\}\} \right.$$
$$\left. + \min\{\alpha A z \mid z \ge 0,\ Az \ge A(x-y)\} \right\}$$
$$- \min_{y \in S_0} \left[d(y) + \min\{\gamma f \mid f = g^0 - Gy\} \right.$$
$$\left. + \min_{x \in S_I(0)} \{\min\{\alpha A(z-y) \mid z \ge y,\ Az \ge Ax\}\} \right].$$

If $x \in S_I(0)$ is now weakened to $x \in S_L(0)$, then

$$z \ge y, \quad Az \ge Ax, \quad x \in S_L(0)$$

is equivalent to the seemingly weaker

$$z \ge y, \quad Az \ge a^0.$$

Since if z satisfies the latter, then $x = z \in S_L(0)$ can be used to give a solution to the former. Hence,

$$\pi_2 = \min_{y \in X_R} \{d(y) + \min\{\gamma f \mid f = g^2 - Gy\} + \min\{\alpha A(z-y) \mid z \in S_2(y)\}\}. \quad (26)$$

For our example, with $\alpha^1 = \alpha^2 = 0$,

$$\pi_2 = \min\left\{\begin{matrix}\gamma & \frac{3}{4} \\ \gamma & \frac{1}{4}\end{matrix}\right\} = \tfrac{1}{4}, \text{ again.}$$

5.4

The case Δ_4 is similar, but we arrive at

$$\begin{aligned}\pi_0 = \min_{y \in X_R} \{d(y) &+ \min\{\gamma f \mid f = g^0 - Gy\} \\ &+ \min\{\alpha(a - Ay) \mid a \geq Ax \geq a^0, \text{ for some } x \geq y\}\},\end{aligned} \quad (27)$$

which can, in turn, be weakened to give

$$\pi_0 = \min_{y \in X_R} \{d(y) + \min\{\gamma f : f = g^0 - Gy\} + \min\{\alpha(a - Ay) : a \geq a^0\}\}. \quad (28)$$

Finally, we note that in (26), (27), and (28), the special property of diamond functions

$$\min\{\gamma f \mid f = g^0 - Gy\} = \sum_{i=1}^{m} \alpha_i(R_i - f_i, \gamma_i^+ f_i, \gamma_i^-(1 - f_i)), \quad \text{where } f = F(g^0 - Gy),$$

is required, in somewhat the same manner as in property P11.

In conclusion, the framework developed here is the construction of a subadditive function π_j based on the generator set X_G, such that $\pi_j \leq c_j$ for all $j = 1, \ldots, n$. The set X_G expands, and this expansion is guided by the candidate set X_C. In this section, several (successively weaker but easier to use) forms of π to determine π_0 have been given beginning with an integer program (24), then a group minimization (25), and finally three linear programs (26), (27), and (28).

6. The algorithmic scheme

The algorithm proceeds in two stages: group enumeration and LP optimization of the parameters γ and α. The more time one spends on parameter optimization, the more the algorithm resembles a cutting plane method, and the more time spent on enumeration, the more it resembles branch and bound, or enumeration. In either extreme, the method is new, and it utilizes the underlying group structure.

The two steps, enumeration and optimization, alternate, but the enumeration step does not require completion of the optimization of the parameters γ and α. On the other hand, completion of the optimization of the parameters must be followed by some amount of enumeration in order to proceed. The enumeration is guided by the outcome of the parameter optimization.

In this section we describe only the *enumeration phase* with fixed γ and α (except for a norming factor α_0). An integer program can be solved with this phase alone but to do so may require an unnecessarily large amount of enumeration. Some optimization (presented in Section 7) should eliminate much enumeration by adjusting the parameters γ and α.

Initially, let the *generator set* X_G be the origin $X_G = \{0\}$. At this initial stage, $X_G = \{y^0\}$ where $y^0_j = 0$, $j = 1, \ldots, n$, and $d(y^0) = 0$. We also introduce the *candidate set* X_C. Initially $X_C = \{\delta^j \mid j = 1, \ldots, n\}$. If any δ^j is known not to belong to X_1, where X_1 from definition 3 is those points $x \in Z^n_+$ with $S_1(x)$ non-empty, then δ^j can be deleted from X_C. More generally, any time an $x \in X_C$ is known to have no optimal solution z to the integer program with $z \geq x$, then x can be deleted from X_C. In this case, the point x never enters the set X_G, and, in common integer programming terminology, one could say that x has been "fathomed". No point larger than x need ever be put in X_C.

There are two readily available means of fathoming an $x \in X_C$. One is the obvious upper bound restrictions. These may be placed in the constraints (1) but can also be imposed here. The second method is by use of the lower bounds on the objective as discussed in Section 6 (or any other method of finding such bounds).

The functions Δ and π will be left unspecified, but Δ_2 or Δ_1 would normally be used. Except for computational difficulties, any subadditive function Δ could be used. Because of our reliance on Theorem 4 and Corollary 1, we are taking $\Gamma(x) = X_G \cap \underline{S}(x)$. Let the parameters γ and α be fixed.

Assume that

$$\min_{x \in S} \{\pi(x)\} > 0$$

for some $S \supseteq S_1(0)$. Then, scale π so that this minimum is equal to one. For $\Delta = \Delta_2$, this assumption becomes

$$1 = \min\{\gamma f \mid f = g^0\} + \min\{\alpha A z \mid z \in S_1(0)\},$$

using (26) with $X_G = \{0\}$.

Let

$$\alpha_0 = \min_{j=1,\ldots,n} \left\{ \frac{c_j}{\Delta(\delta^j)} \,\middle|\, \Delta(\delta^j) > 0 \right\}.$$

Then $\alpha_0 > 0$ since $c_j > 0$. The initial subadditive function π is given by $\pi(x) = \alpha_0 \Delta(x)$ and the initial bound on the objective is α_0.

The general step of the enumeration is to find $x^* \in X_C$ for which

$$\frac{d(x^*)}{\Delta(x^*)} = \min\left\{ \frac{d(x)}{\Delta(x)} \,\middle|\, x \in X_C \quad \text{and} \quad \Delta(x) > 0 \right\}.$$

Change α_0 to

$$\alpha_0 - \frac{d(x^*)}{\Delta(x^*)},$$

move x^* from X_F to X_E and adjoin to X_F those points $x > x^*$, $x \in Z^n_+$ such that every $y < x$, $y \in Z^n_+$ has $y \in X_E$. In addition, we can drop any such x from X_F if it is known to not belong to X_0. For instance, x can be dropped if $x \notin X_1 \supseteq X_0$ which is easily checked. Because of the norming factor α_0, the bound π_0 is always given by

$$\pi_0 = \min_{y \in X_E} \left\{ d(y) + \min_{x \in S} \alpha_2 \Delta(x - y) \right\}$$

for any $S \supseteq S_F(0)$. If Δ_2 is used, then the bound is given by (26); that is,

$$\pi_0 = \min_{y \in X_E} \left\{ d(y) + \alpha_2 \left[\sum_{i=1}^{m} \sigma(R_i - f_i, \gamma_i^+ f_i, \gamma_i^-(1 - f_i)) + \min \{\sigma A(z - y) \mid z \in S_F(y)\} \right] \right\}$$

where $f = F(g^0 - Gy)$ and R_i is as in property P11. The minimum over $\sigma A z$ is a separable, convex linear programming problem of the form discussed in property P10. For each $y \in X_E$ this minimization need only be solved once since only α_0 is being varied here.

The computation $\Delta(x)$, $x \in X_F$, need only be done once for each such x since Δ is not changed in this enumerative phase of the algorithm with fixed γ and σ.

We illustrate the algorithm using the example previously introduced. Beginning with $\gamma^+ = \gamma^- = \frac{1}{4}$, $\alpha^+ = \alpha^- = 0$, the initial bound is $\pi_0 = \frac{1}{2}$. Now, let X_E be enlarged to $X_E = \{(0,0), (1,0), (0,1)\}$. Then, $X_F = \{(2,0), (1,1), (0,2)\}$ and

$$\Delta_2(2,0) = \min \begin{Bmatrix} [illegible] \\ [illegible] \end{Bmatrix} = [illegible],$$

$$\Delta_2(1,1) = \min \{[illegible] \times 0\} = 0,$$

$$\Delta_2(0,2) = \min \begin{Bmatrix} [illegible] \\ [illegible] \end{Bmatrix} = [illegible].$$

Hence, α_2 can be raised to 4, since $d(x) = 2$ for each $x \in X_F$. The new π_0 is given by

$$\pi_0 = \min \{0 + 4 \times \tfrac{1}{2}, 1 + 4 \times [illegible] \times [illegible], 1 + 4 \times [illegible] \times [illegible]\} = 2.$$

Both (2,0) and (0,2) should be put in X_E. Then $X_F = \{(3,0), (0,3), (1,1)\}$ and

$$\Delta_2(3,0) = 4 \times [illegible] \times [illegible] = 2$$

$$\Delta_2(0,3) = 4 \times [illegible] \times [illegible] = 1.$$

Thus, α_2 can be increased to 6, since both (3,0) and (0,3) have $d = 3$. However, π_0 remains at 2 because for $y = (0,2) \in X_E$, $g^0 - G_y = 0$ and $d(y) = 2$. There are two ways to proceed from here. One is to simply delete (0, 2) from X_E because any $z \geq (0,2)$ must violate $-4z_1 - 11z_2 \geq -12$. The other is to adjust the parameters $\gamma^+, \gamma^-, \alpha^+, \alpha^-$ as discussed in the next section. We pick up the discussion of this example there.

7. The parameters γ and α

For a fixed X_E and free parameters γ and α, the problem of finding the best bound π_0 becomes:

$$\max \ \pi_0 \tag{29}$$

$$\text{s.t.} \quad \pi_0 \leq d(y) + \Delta(x-y), \quad y \in X_E, \quad x \in S, \tag{30}$$

$$\Delta(x) \leq d(x), \quad x \in X_E, \tag{31}$$

where $S \supseteq S_1(y)$ and Δ could be any of the $\Delta_0, \Delta_1, \Delta_2, \Delta_3$ functions.

We first must remark on the use of X_E or X_E^*. In Theorem 4, $d(x) \geq \Delta(x)$ is required on X_E^*. However, $X_E^* \subset (X_E \cup X_H)$ and so if $x \in X_E^*$, $x \notin X_E$, then $x \in X_H$, and consequently every $y \leq x$ has $d(y) \geq \Delta(y)$. Further $d(x) \geq \Delta(x)$. Hence, (31) is satisfied. If $x \in X_E^*$ and $x \in X_E$ then clearly (31) is satisfied. Of course the bound π_0 given by (30) would be improved if $X_E^* \subset X_E$ was used. However, it is more convenient to not insist on keeping X_E as small as possible and to only contract it to X_E^* occasionally if at all.

Using $\Delta = \Delta_2$, the constraint (31) becomes

$$d(x) \geq \sum_{i=1}^{m_1} \sigma(R_i - f_i, \gamma_i^+ f_i, \gamma_i^-(1-f_i)) + \min\{\alpha A z \mid z \geq 0, Az \geq Ax\}, \quad x \in X_E,$$

where $f = F(Gx)$, and $R_i = \gamma_i^-/(\gamma_i^+ + \gamma_i^-)$. The minimization of $\alpha A z$ can be simplified by taking $z = x$ instead of the optimal z, thereby weakening the bound. Then, (31) becomes

$$d(x) \geq \sum_{i=1}^{m_1} \sigma(R_i - f_i, \gamma_i^+ f_i, \gamma_i^-(1-f_i)) + \sum_{i=1}^{m_2} b_i \sigma(b_i, \alpha_i^+, \alpha_i^-),$$

where $b = Ax$. This constraint is linear in γ and α except for the R term, which is a rational function of γ. If for each i we require γ_i^+, γ_i^- to satisfy either

$$\gamma_i^+ f_i \leq \gamma_i^-(1-f_i), \text{ thus } R_i - f_i \geq 0,$$

or

$$\gamma_i^+ f_i \geq \gamma_i^-(1-f_i), \text{ thus } R_i - f_i \leq 0,$$

for all $f = Gx$, $x \in X_E$, then a certain region of γ_i^+, γ_i^- will be allowed as shown in Fig. 2. These regions may be modified during the course of the algorithm to improve the bound π_0. The constraints on γ_i^+, γ_i^- are linear once the choice of $R_i - f_i \geq 0$ or $R_i - f_i \leq 0$ is made for each $f = Gx$, $x \in X_E$.

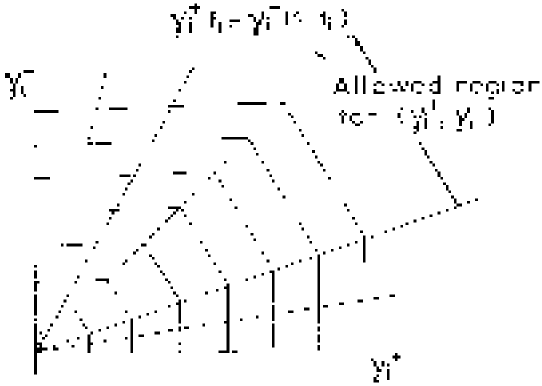


Fig. 2

The constraints (30) can be written, for $A = A_\mu$, using (26) as

$$\pi_E \le d(y) + \sum_{i=1}^{m_1} \alpha_i (R_i - f_i \gamma_i^+ f_i - \gamma_i^-(1 - f_i)) - \min\{\alpha A(z - y) \mid z \in S_F(y)\}, \quad y \in X_E,$$

where $f = F(g^- - Gy)$. As is true for the constraints (31), this constraint may be reduced to a linear constraint by restricting γ_i^+, γ_i^- to an adequately specified region of $\mathbf{R}^2$.

The minimization over $\alpha A z$ is a linear program with separable, piecewise linear, convex objective, for a given α. By solving each such problem for $y \in X_E$ and then adjoining linear restrictions on α to assure that those solutions remain optimal, the parameter optimization can be kept as a linear problem.

At this point, we proceed with the example using (28). Table 1 represents $X_F = \{(0,0),(1,0),(0,1),(2,0),(0,2)\}$ and $X_E = \{(3,0),(1,1),(0,3)\}$. If we solve the indicated linear program, we find $\gamma^+ = 10\frac{2}{3}$, $\gamma^- = 7$, $\alpha^+ = \frac{1}{27}$, $\alpha^- = -\frac{1}{18}$, and $\pi_E = 3$. In fact, the linear program picks out the optimum answer (3, 0).

Table 1

	X_F				X_E									
y_1	2	0	0	−[illegible]	0	[illegible]	0	0	[illegible]	[illegible]	0	0	—	0
y_2	0	0	[illegible]	0	[illegible]	0	[illegible]	0	−[illegible]	[illegible]	0	0	—	0
π	0	0	0	0	0	0	[illegible]	10	0	0	1	1	—	0
α'	12	15	32	−12	8	−1	−4	0	0	0	−1	1		0
[illegible]	0	0	0	1	1	1	[illegible]	1	0	0	0	0	—	1
d	2	2	3	0	1	1	2	2	0	0	0	0		

References

[1] J. Araoz, Polyhedral neopolarities, Thesis, University of Waterloo, November 1973.
[2] D. Bell, The resolution of duality gaps in discrete optimization, Tech Report No. 81, Operations Research Center, MIT, 1973.

[3] C.-A. Burdet, Enumerative cuts I, *Operations Res.* 21 (1973) 61–89.

[4] C.-A. Burdet, On the algebra and geometry of integer programming cuts, Management Sciences Report No. 291, Carnegie-Mellon University, Pittsburgh, PA., October 1972.

[5] C.-A. Burdet and E.L. Johnson, A subadditive approach to the group problem of integer programming, *Math. Programming Study* 2 (1974) 51–71.

[6] M.L. Fisher, W.D. Northup and J.F. Shapiro, Using duality to solve discrete optimization problems: Theory and computational experience, Working Paper OR 030-74, Operations Research Center, M.I.T., 1974.

[7] R.E. Gomory, Some polyhedra related to combinatorial problems, *Linear Algebra and Its Appl.* 2 (1969) 451–558.

[8] R.E. Gomory and E.L. Johnson, Some continuous functions related to corner polyhedra I, *Math. Programming* 3 (1972) 23–85.

[9] E.L. Johnson, Cyclic groups, cutting planes, and shortest paths, in: T.C. Hu and S. Robinson, eds., *Mathematical Programming* (Academic Press, New York, 1973).

[10] E.L. Johnson, On the group problem for mixed integer programming, *Math. Programming Study* 2 (1974) 137–179.

[11] E.L. Johnson, Integer programs with continuous variables, Institute for Econometrics and Operations Research, University of Bonn, Nassestrasse 2, West Germany, Report No. 7419-OR, September 1974.

[12] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, N.J., 1970).

Annals of Discrete Mathematics 1 (1977) 145–164

AGGREGATION OF INEQUALITIES IN INTEGER PROGRAMMING*

Václav CHVÁTAL
Dept. of Computer Science, Stanford University, Stanford, CA 94305, U.S.A.

Peter L. HAMMER
Dept. of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Given an $m \times n$ zero-one matrix A, we ask whether there is a single linear inequality $ax \leq b$ whose zero-one solutions are precisely the zero-one solutions of $Ax \leq e$. We develop an algorithm for answering this question in $O(mn^2)$ steps and investigate other related problems. Our results may be interpreted in terms of graph theory and threshold logic.

1. Introduction

Given a set of linear equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad (i = 1, 2, \ldots, m), \tag{1.1}$$

one may ask whether there is a single linear equation

$$\sum_{j=1}^{n} a_j x_j = b \tag{1.2}$$

such that (1.1) and (1.2) have precisely the same set of zero-one solutions. As shown by Bradley [2], the answer is always affirmative. (Actually, Bradley's results are more general. Some of them have been generalized further by Rosenberg [12].) In this paper, we shall consider a related question: given a set of linear inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad (i = 1, 2, \ldots, m), \tag{1.3}$$

we shall ask whether there is a single linear inequality

$$\sum_{j=1}^{n} a_j x_j \leq b \tag{1.4}$$

such that (1.3) and (1.4) have precisely the same set of zero-one solutions. In a sense, which we are about to outline, this problem has been solved long ago.

First, a few definitions. A function

* This research was partly carried out at the Centre de recherches mathématiques, Université de Montréal, partial support of the NRC (Grant A 8552) is gratefully acknowledged.

$$f : \{0,1\}^n \to \{0,1\}$$

is called a *switching function*. If there are real numbers $a_1, a_2, \ldots, a_n$ and b such that

$$f(x_1, x_2, \ldots, x_n) = 0 \iff \sum_{j=1}^{n} a_j x_j \le b$$

then f is called a *threshold function*. If there are (not necessarily distinct) zero-one vectors $y_1, y_2, \ldots, y_k$ and $z_1, z_2, \ldots, z_k$ such that

$$f(y_i) = 0, \qquad f(z_i) = 1, \quad \text{for all } i = 1, 2, \ldots, k,$$

$$\sum_{i=1}^{k} y_i = \sum_{i=1}^{k} z_i,$$

then, for each integer m with $m \ge k$, the function f is called *m-summable*. If f is not m-summable then f is called *m-assumable*. It is well-known [3, 8] that a switching function is threshold of and only if it is m-assumable for every m. (The proof is quite easy: denote by S_i the set of all the zero-one vectors x with $f(x) = i$. By definition, f is threshold if and only if there is a hyperplane separating S_0 from S_1. Such a hyperplane exists if and only if the convex hulls of S_0 and S_1 are disjoint. Clearly, these convex hulls are disjoint if and only if f is m-assumable for every m.)

Coming back to our problem, we may associate with (1.3) a switching function f defined by

$$f(x_1, x_2, \ldots, x_n) = 0 \iff (1.3) \text{ holds.}$$

Then the desired inequality (1.4) exists if and only if f is m-assumable for every m. However, such an answer to our question is unsatisfactory on several counts. Above all, it does not provide an *efficient* algorithm for deciding whether (1.4) exists. We shall develop such an algorithm in the special case when all the coefficients a_{ij} and b_i in (1.3) are zeroes and ones.

An $m \times n$ zero-one matrix $A = (a_{ij})$ will be called *threshold* if, and only if, there is a single linear inequality

$$\sum_{j=1}^{n} a_j x_j \le b$$

whose zero-one solutions are precisely the zero-one solutions of the system

$$\sum_{j=1}^{n} a_{ij} x_j \le 1 \quad (i = 1, 2, \ldots, m). \tag{1.5}$$

Note that the zero-one solutions of (1.5) are completely determined by the set of those pairs of columns of A which have a positive dot product. This information is conveniently described by means of a graph; in order to make our paper self-contained, we shall now present a few elementary definitions from graph theory.

A *graph* G is an ordered pair (V, E) such that V is a finite set and E is some set of two-element subsets of V. The elements of V are called the *vertices* of G, the elements of E are called the *edges* of G. Two vertices $u, v \in V$ are called *adjacent* if $\{u, v\} \in E$ and *nonadjacent* otherwise. For simplicity, we shall denote each edge $\{u, v\}$ by uv. A subset S of V is called *stable* in G if no two vertices from S are adjacent in G.

With each $m \times n$ zero-one matrix A, we shall associate its *intersection graph* $G(A)$ defined as follows. The vertices of $G(A)$ are in a one-to-one correspondence with the columns of A; two such vertices are adjacent if and only if the corresponding columns have a positive dot product. The motivation for introducing the concept is obvious: the zero-one solutions of (1.5) are precisely the characteristic vectors of stable sets in $G(A)$. We shall call a graph G with vertices $u_1, u_2, \ldots, u_n$ *threshold* if there are real numbers $a_1, a_2, \ldots, a_n$ and b such that the zero-one solutions of

$$\sum_{j=1}^{n} a_j x_j \leq b$$

are precisely the characteristic vectors of stable sets in G. Clearly, $G(A)$ is threshold if and only if A is threshold; let us also note that $G(A)$ can be constructed from A in $O(mn^2)$ steps. Thus the question "Is A threshold?" reduces into the question "Is $G(A)$ threshold?".

2. The main result

In this section, we develop an algorithm for deciding, within $O(n^2)$ steps, whether a graph G on n vertices is threshold. We shall begin by showing that certain small graphs are not threshold. These graphs are called $2K_2$, P_4 and C_4; they are shown in Fig. 1.

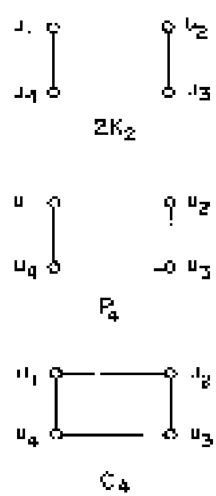


Fig. 1.

Fact 1. *If G is $2K_2$, P_4 or C_4, then G is not threshold.*

Proof. Assume that one of the above graphs G is threshold. Then there is a linear inequality

$$a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \le b$$

whose zero-one solutions are precisely the characteristic vectors of stable sets in G. In particular, we have

$$a_1 + a_2 > b, \qquad a_3 + a_4 > b, \qquad a_1 + a_3 \le b, \qquad a_2 + a_4 \le b,$$

clearly, these four inequalities are inconsistent. $\square$

In order to make our next observation about threshold graphs, we need the notion of an "induced subgraph". Let $G = (V, E)$ be a graph and let S be a subset of V. The subgraph of G *induced* by S is the graph H whose set of vertices is S; two such vertices are adjacent in H *if and only if* they are adjacent in G.

Fact 2. *If G is a threshold graph, then every induced subgraph of G is threshold.*

Proof. Let the zero-one solutions of

$$\sum_{j=1}^{n} a_jx_j \le b$$

be precisely the characteristic vectors of stable sets in G. Let H be a subgraph of G induced by S. Denote by $\sum^*$ the summation over all the subscripts j with $u_j \in S$. Then the zero-one solutions of

$$\sum{}^* a_jx_j \le b$$

are precisely the characteristic vectors of stable sets in H. $\square$

Now, we have an easy way of showing that certain graphs are not threshold (simply by pointing out an induced subgraph isomorphic to $2K_2$, P_4 or C_4). On the other hand, we are about to develop a way of showing that certain graphs *are* threshold. Let G be a graph with vertices $u_1, u_2, \ldots, u_n$. G will be called *strongly threshold* if there are *positive integers* $a_1, a_2, \ldots, a_n$ and b such that the zero-one solutions of

$$\sum_{j=1}^{n} a_jx_j \le b$$

are precisely the characteristic vectors of stable sets in G. (It will turn out later, and may be proved directly, that every threshold graph is strongly threshold.) We shall show that the property of being strongly threshold is preserved under two simple operations. Let G be a graph with vertices $u_1, u_2, \ldots, u_n$. By $G + K_1$, we shall denote the graph obtained from G by adding a new vertex u_{n+1} and all the edges

v_iv_{n+1} with $1 \le i \le n$. $G \cup K_1$, we shall denote the graph obtained from G by adding a new vertex v_{n+1} and no edges at all.

Fact 3. *If G is strongly threshold then $G + K_1$ and $G \cup K_1$ are strongly threshold.*

Proof. Let $a_1, a_2, \ldots, a_n$ and b be positive integers such that the zero-one solutions of

$$\sum_{i=1}^{n} a_ix_i \le b$$

are precisely the characteristic vectors of stable sets in G. Then the zero-one solutions of

$$\sum_{i=1}^{n} a_ix_i + bx_{n+1} \le b$$

are precisely the characteristic vectors of stable sets in $G + K_1$. Similarly, the zero-one solutions of

$$2\sum_{i=1}^{n} a_ix_i + x_{n+1} \le 2b + 1$$

are precisely the characteristic vectors of stable sets in $G \cup K_1$. $\square$

Now, we are ready for the theorem.

Theorem 1. *For every graph G, the following three conditions are equivalent:*

(i) *G is threshold,*

(ii) *G has no induced subgraph isomorphic to $2K_2$, P_4 or C_4,*

(iii) *there is an ordering $v_1, v_2, \ldots, v_n$ of the vertices of G and a partition of $\{v_1, v_2, \ldots, v_n\}$ into disjoint subsets P and Q such that*

every $v_j \in P$ is adjacent to all the vertices v_i with $i < j$,

every $v_j \in Q$ is adjacent to none of the vertices v_i with $i < j$.

Proof. The implication (i) $\Rightarrow$ (ii) follows from Fact 1 and Fact 2. The implication (iii) $\Rightarrow$ (i) may be deduced from Fact 3. Indeed, let G_t denote the subgraph of G induced by $\{v_1, v_2, \ldots, v_t\}$. If $v_{t+1} \in P$ then $G_{t+1} = G_t + K_1$; if $v_{t+1} \in Q$, then $G_{t+1} = G_t \cup K_1$. Hence, by induction on t, every G_t is strongly threshold.

It remains to be proved that (ii) $\Rightarrow$ (iii). We shall accomplish this by means of an algorithm which finds, for every graph G, either one of the three forbidden induced subgraphs or the ordering and partition described in (iii). If G has n vertices then the algorithm takes $O(n^2)$ steps.

Before the description of the algorithm, a few preliminary remarks may be in order. It will be convenient to introduce the notion of the *degree* $d_G(u)$ of a vertex u in a graph G; this quantity is simply the number of vertices of G which are adjacent to u. At each stage of the algorithm, we shall deal with some sequence S of k

vertices of G; the remaining vertices will already be enumerated as $v_{k+1}, v_{k+2}, \ldots, v_n$ and partitioned into sets P and Q. Furthermore, each $w \in S$ will be adjacent to all the vertices from P and to no vertices from Q; hence it will be adjacent to exactly $d_G(w) - |P|$ vertices from S.

Step 0. For each vertex w of G, evaluate $d_G(w)$. (This may take as many as $O(n^2)$ steps.) Then arrange the vertices of G into a sequence $w_1, w_2, \ldots, w_n$ such that

$$d_G(w_1) \geq d_G(w_2) \geq \cdots \geq d_G(w_n);$$

call this sequence S. (This can be done in $O(n \log n)$ steps; the rest of the algorithm takes only $O(n)$ steps.) Set $k = n$ and $P = Q = \emptyset$.

Step 1. If $k = 1$, then S has only one term; call that vertex v_1 and stop. If $k > 1$ then let u be the first term of S and let v be the last term of S; note that

$$|P| + k - 1 \geq d_G(u) \geq d_G(w) \geq d_G(v) \geq |P|$$

for every $w \in S$. If $d_G(u) = |P| + k - 1$, go to Step 2. If $d_G(v) = |P|$, go to Step 3. If $|P| < d_G(v) \leq d_G(u) < |P| + k - 1$, go to Step 4.

Step 2. Set $v_k = u$, delete u from S, replace P by $P \cup \{v_k\}$, replace k by $k - 1$ and return to Step 1.

Step 3. Let $v_k = v$, delete v from S, replace Q by $Q \cup \{v_k\}$, replace k by $k - 1$ and return to Step 1.

Step 4. Let $u_1 = u$. Find a vertex $u_2 \in S$ which is not adjacent to u_1. Find a vertex $u_3 \in S$ which is adjacent to u_2. Find a vertex $u_4 \in S$ which is adjacent to u_1 but not to u_3. Then stop (the vertices u_1, u_2, u_3, u_4 induce $2K_2$ or P_4 or C_4 in G). □

Remark. In Step 4, we take the existence of u_4 for granted. However, the existence of such a vertex follows at once from the fact that $d_G(u_1) \geq d_G(u_3)$ and from the fact that u_3 is adjacent to u_2 but not to u_1.

In the rest of this section, we shall present several consequences of Theorem 1.

Remark 1. For every graph $G = (V, E)$, we may define a binary relation $\preccurlyeq$ on V by writing $u \preccurlyeq v$ if and only if

$$uw \in E,\ w \neq v \quad \Rightarrow \quad vw \in E.$$

By this definition, $\preccurlyeq$ is reflexive and transitive but not necessarily antisymmetric.

From Theorem 1, we conclude the following.

Corollary 1A. *A graph G is threshold if and only if for every two distinct vertices u, v of G, at least one of $u \preccurlyeq v$ and $v \preccurlyeq u$ holds.*

Remark 2. For every graph $G = (V, E)$ and for every vertex u of G, we define

$$N(u) = \{v \in V : v \text{ is adjacent to } u\}.$$

From Theorem 1, we conclude the following.

Corollary 1B. *A graph $G = (V, E)$ is threshold if and only if there is a partition of V into disjoint sets A, B and an ordering $u_1, u_2, \ldots, u_a$ of A such that*

no two vertices in A are adjacent,

every two vertices in B are adjacent,

$N(u_1) \subseteq N(u_2) \subseteq \cdots \subseteq N(u_a)$.

Let us sketch the proof. If G has the structure described by Corollary 1B then G cannot possibly have an induced subgraph isomorphic to $2K_2$, P_4 or C_4; hence G is threshold. On the other hand, if G is threshold then G has the structure described by (iii) of Theorem 1. In that case, we may set $A = Q$, $B = V - Q$. Finally, we scan the list $v_1, v_2, \ldots, v_k$ in the *reverse* order (from v_k to v_1) and enumerate the vertices of B as $u_1, u_2, \ldots, u_a$.

Remark 3. For every graph G, we define the *complement* $\bar{G}$ of G to be a graph with the same set of vertices as G; two distinct vertices are adjacent in $\bar{G}$ if and only if they are not adjacent in G. From the equivalence of (i) and (ii) in Theorem 1, we conclude the following.

Corollary 1C. *A graph is threshold if and only if its complement is threshold.*

Let us point out that this fact does not seem to follow directly from the definition.

Remark 4. In order to decide whether a graph G (with vertices $u_1, u_2, \ldots, u_n$) is threshold, it suffices to know only the degrees $d_G(u_1), d_G(u_2), \ldots, d_G(u_n)$ of its vertices. Indeed, executing Steps 1, 2 and 3 of the algorithm, we manipulate only these quantities. On the other hand, if we are about to execute Step 4 then we already know that G is not threshold.

Remark 5. Theorem 1 implies that threshold graphs are very rare. Indeed, from (iii) of Theorem 1, we conclude that the number of distinct threshold graphs with vertices $u_1, u_2, \ldots, u_n$ does not exceed

$$n!\,2^{n-1}.$$

On the other hand, the number of all distinct graphs with the same set of vertices is

$$2^{n(n-1)/2}.$$

Hence a randomly chosen graph will almost certainly be not threshold.

Remark 6. With each graph G on vertices $u_1, u_2, \ldots, u_n$ we may associate a switching function

$$f: \{0, 1\}^n \to \{0, 1\}$$

by setting $f(x_1, x_2, \ldots, x_n) = 0$ if and only if $(x_1, x_2, \ldots, x_n)$ is the characteristic vector of some stable set in G. A switching function arising in this way will be called *graphic*.

From Theorem 1, we conclude the following.

Corollary 1D. *A graphic switching function is threshold if and only if it is 2-assumable.*

Let us point out that for switching functions that are *not* graphic, the "if" part of Corollary 1D is no longer true. Indeed, for every m with $m \geq 2$, there are switching functions which are m-assumable but not $(m+1)$-assumable. Ingenious examples of such functions have been constructed by Winder [14].

Remark 7. When $A = (a_{ij})$ is an $m \times n$ zero-one matrix, we shall consider the following zero-one linear programming problem:

$$
\begin{aligned}
&\text{maximize} && \sum_{j=1}^{n} c_j x_j \\
&\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad (1 \leq i \leq m), \\
& && x_j = 0, 1 \qquad (1 \leq j \leq n).
\end{aligned}
\tag{2.1}
$$

Defining $c(v_j) = c_j$ for every vertex v_j of $G(A)$, we reduce (2.1) to the following problem:

$$
\begin{aligned}
&\text{in } G(A), \text{ find a stable set } S \\
&\text{maximizing } c(S) = \sum_{v \in S} c(v).
\end{aligned}
\tag{2.2}
$$

In general, (2.2) is hard; one may ask whether it becomes any easier when A is threshold. The answer is affirmative. Indeed, if $G(A)$ is threshold then we can find the ordering $v_1, v_2, \ldots, v_n$ and the partition $P \cup Q$ described in (iii), Theorem 1; this takes only $O(mn^2)$ steps. Then we define

$$
S_1 = \begin{cases} \emptyset & \text{if } c(v_1) < 0, \\ \{v_1\} & \text{if } c(v_1) \geq 0 \end{cases}
$$

and, for each t with $2 \leq t \leq n$,

$$
S_t = \begin{cases} S_{t-1} & \text{if } v_t \in Q \text{ and } c(v_t) < 0, \\ S_{t-1} \cup \{v_t\} & \text{if } v_t \in Q \text{ and } c(v_t) \geq 0, \\ S_{t-1} & \text{if } v_t \in P \text{ and } c(v_t) < c(S_{t-1}), \\ \{v_t\} & \text{if } v_t \in P \text{ and } c(v_t) \geq c(S_{t-1}). \end{cases}
$$

Clearly, S_0 is a solution of (2.2).

Remark 8. G. Minty observed that it is quite easy to decide whether a threshold graph has a hamiltonian cycle. We reproduce his observation below with a slightly different proof. A graph G is called 1-tough if for every nonempty set S of its vertices, the graph $G - S$ has at most $|S|$ components.

Corollary 1E. *For every threshold graph G, the following three conditions are equivalent:*

(i) *G is hamiltonian.*

(ii) *G is 1-tough.*

(iii) *the degree sequence $d_1 \le d_2 \le \cdots \le d_n$ of G is such that*

$$d_j \le j < n/2 \implies d_{n-j} \ge n - j.$$

Proof. Since the implications (i) $\Rightarrow$ (ii) and (iii) $\Rightarrow$ (i) hold for arbitrary graphs (see [4, 5]), it suffices to prove that (ii) $\Rightarrow$ (iii). For this purpose, we shall consider a threshold graph G violating (iii) and prove that G violates (ii). Thus we have $d_j \le j$ and $d_{n-j} < n - j$ for some $j < n/2$. Representing G as in Corollary 1B, we observe that $d_i = d_G(u_i)$ for $i = 1, 2, \dots, k$. Let us distinguish two cases.

Case 1: $j \le k$. Define $S = N(\{u_1, u_2, \dots, u_j\})$ and observe that $|S| = d_j \le j < n/2$. Note also that $G - S$ contains j isolated vertices $u_1, u_2, \dots, u_j$ and at least one additional component. Hence G is not 1-tough.

Case 2: $j > k$. This case simply cannot occur since $d_j \ge d_{k+1} \ge n - k - 1 \ge n - j$.

3. Variations

Let $A = (a_{ij})$ be an $m \times n$ zero–one matrix. We shall denote by $t(A)$ the smallest t for which there exists a system of linear inequalities

$$\sum_{j=1}^{n} c_{ij}x_j \le d_i \quad (1 \le i \le t) \tag{3.1}$$

such that (3.1) and

$$\sum_{j=1}^{n} a_{ij}x_j \le 1 \quad (1 \le i \le m) \tag{3.2}$$

have the same set of zero–one solutions. Theorem 1 characterizes matrices A with $t(A) = 1$; in this section, we shall discuss the problem of finding $t(A)$ for every matrix A.

Again, the language of graph theory will be useful. For every graph $G = (V, E)$, we shall denote by $t(G)$ the smallest t such that there are threshold graphs $G_1 = (V, E_1), G_2 = (V, E_2), \dots, G_t = (V, E_t)$ with $E_1 \cup E_2 \cup \cdots \cup E_t = E$. Our next result may not sound too surprising. Note, however, that Theorem 1 is used in its proof.

Theorem 2. *Let A be a zero-one matrix and let G be $G(A)$. Then $t(A) = t(G)$.*

Proof. The inequality $t(A) \le t(G)$ is fairly routine. Indeed, there are t threshold graphs $G_i = (V, E_i)$ with $\bigcup E_i = E$ and $t = t(G)$. For each i there is an inequality

$$\sum_{j=1}^{n} c_{ij} x_j \le d_i$$

whose zero-one solutions are precisely the characteristic vectors of stable sets in G_i. A subset of V is stable in G if and only if it is stable in every G_i. Hence the zero-one solutions of the system

$$\sum_{j=1}^{n} c_{ij} x_j \le d_i \quad (1 \le i \le t) \tag{3.3}$$

are precisely the characteristic vectors of stable sets in G. Since $G = G(A)$, the characteristic vectors of stable sets in G are precisely the zero-one solutions of (3.1). Hence $t(A) \le t = t(G)$.

In order to prove the reversed inequality, we shall use Theorem 1. There is a system (3.2) with $t = t(A)$ such that (3.1) and (3.2) have the same set of zero-one solutions. Set $V = \{u_1, u_2, \dots, u_n\}$; for each i, define

$$E_i = \{u_r u_s : r \ne s \text{ and } c_{ir} + c_{is} > d_i\}$$

and $G_i = (V, E_i)$. Since (3.1) and (3.2) have the same set of zero-one solutions, we have

$$\bigcup_{i=1}^{t} E_i = \{u_r u_s : a_{kr} + a_{ks} > 1 \quad \text{for some } k = 1, 2, \dots, m\}.$$

Hence $G = (V, \bigcup E_i)$ is $G(A)$; it remains to be proved that each G_i is threshold. Assume the contrary. Then, by part (ii) of Theorem 1, there are vertices u_p, u_q, u_r, u_s such that

$$u_s u_q \in E_i, \qquad u_r u_p \in E_i,$$

$$u_r u_q \notin E_i, \qquad u_s u_p \notin E_i.$$

Hence by the definition of E_i, we have

$$c_{is} + c_{iq} > d_i, \qquad c_{ir} + c_{ip} > d_i,$$

$$c_{ir} + c_{iq} \le d_i, \qquad c_{is} + c_{ip} \le d_i.$$

Clearly, these four inequalities are inconsistent. $\square$

Next, we shall establish an upper bound on $t(G)$. In order to do that, we shall need a few more graph-theoretical concepts. A *triangle* is a graph consisting of three pairwise adjacent vertices; a *star* (centered at u) is a graph all of whose edges contain the same vertex u. The *stability number* $\alpha(G)$ of a graph G is the size of the largest stable set in G.

Theorem 3. *For every graph G on n vertices, we have $t(G) \leq n - \alpha(G)$. Furthermore, if G contains no triangle, then $t(G) = n - \alpha(G)$.*

Proof. Write $G = (V, E)$ and $k = n - \alpha(G)$. Let S be a largest stable set in G; enumerate the vertices in $V - S$ as $u_1, u_2, \ldots, u_k$. For each i with $1 \leq i \leq k$, let E_i consist of all the edges of G which contain u_i. Then each $G_i = (V, E_i)$ is a star and therefore a threshold graph. Since S is stable, we have $\bigcup E_i = E$. Hence $t(G) \leq k$.

Secondly, let us assume that G contains no triangle. There are t threshold graphs $G_i = (V, E_i)$ with $1 \leq i \leq t$, $t = t(G)$ and $\bigcup E_i = E$. It follows easily from Theorem 1 that each G_i, being threshold and containing no triangle, must be a star. Hence there are vertices $u_1, u_2, \ldots, u_t$ such that every edge of every G_i contains u_i. Since $\bigcup E_i = E$, the set

$$V - \{u_1, u_2, \ldots, u_t\}$$

is stable in G. Hence $\alpha(G) \geq n - t(G)$. □

Let us note that we may have $t(G) = n - \alpha(G)$ even when G does contain a triangle. For example, see the graph in Fig. 2.

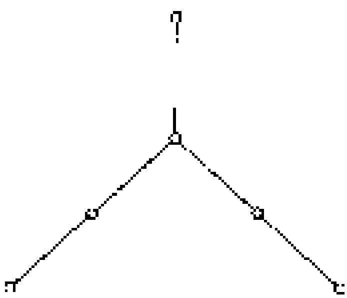


Fig. 2.

When $\alpha(G)$ is very large, the upper bound on $t(G)$ given by Theorem 3 is much smaller than n. On the other hand, if $\alpha(G)$ is very small, then $t(G)$ is often very small. (In particular, if $\alpha(G) = 1$, then $t(G) = 1$.) Thus one might hope that, say, $t(G) \leq n/2$ for every graph on n vertices. Our next result shows such hopes to be very much unjustified.

Corollary 3A. *For every positive ε there is a graph G on n vertices such that $t(G) \geq (1 - \varepsilon)n$.*

Proof. Erdős [9] has proved that for every positive integer k there is a graph G on n vertices such that G contains no triangle, $\alpha(G) < k$ and, for some positive constant c (independent of k), $n \geq c(k/\log k)^2$. Given a positive ε, choose k large enough, so that $\varepsilon ck \geq (\log k)^2$, and consider the graph G with the above properties. We have

$$\alpha(G) \le k \le \frac{n}{ck}(\log k)^2 \le \varepsilon n$$

and so, by Theorem 3, $t(G) = n - \alpha(G) \ge (1-\varepsilon)n$. □

Finally, we shall show that the problem of finding $t(G)$ is very hard; more precisely, we shall show that it is "NP-hard". Perhaps a brief sketch of the meaning of this term is called for. There is a certain wide class of problems; this class is called NP. It includes some very hard problems such as the problem of deciding whether the vertices of a graph are colorable in k colors. An algorithm for solving a problem is called *good* if it terminates within a number of steps not exceeding some (fixed) polynomial in the length of the input [7]. A few years ago, Cook [6] proved that the existence of a good algorithm for finding the stability number of a graph would imply the existence of a good algorithm for *every* problem in NP. Such a conclusion, if true, is very strong. (For example, it implies the existence of a good algorithm for the celebrated traveling salesman problem.) A problem X is called *NP-hard* if the existence of a good algorithm for X would imply the existence of a good algorithm for every problem in NP. (For more information on the subject, the reader is referred to [1] and [10].)

Corollary 3B. *The problem of finding $t(G)$ is NP-hard.*

Proof. Poljak [11] proved that, even for graphs G that contain no triangles, the problem of finding $\alpha(G)$ is NP-hard. For such graphs, however, we have $\alpha(G) = n - t(G)$; hence the existence of a good algorithm for finding $t(G)$ would imply the existence of a good algorithm for Poljak's problem. Since Poljak's problem is NP-hard, our problem is NP-hard. □

We shall close this section with two remarks on $t(G)$.

Remark 1. First of all, we shall present a simple lower bound on $t(G)$. For every graph $G = (V, E)$, let us define a new graph $G^* = (V^*, E^*)$ as follows. The vertices of G^* are the edges of G; that is, $V^* = E$. Two vertices of G^*, say $\{u, v\} \in V^*$ and $\{w, z\} \in V^*$, are adjacent in G^* if and only if the set $\{u, v, w, z\}$ induces $2K_2$, P_4 or C_4 in G. Fig. 2 shows an example of G and G^*.

As usual, the chromatic number $\chi(H)$ of a graph $H = (V, E)$ is the smallest k such that V can be partitioned into k stable sets. We claim that

$$t(G) \ge \chi(G^*). \tag{3.4}$$

Indeed, there are threshold graphs $G_i = (V, E_i)$ with $1 \le i \le t$, $t = t(G)$ and $\bigcup E_i = E$. By (ii) of Theorem 1 and by our definition of G^*, each E_i is a stable set of vertices in G^*. Hence $\chi(G^*) \le t$.

Note that the problem of finding the chromatic number of a graph is NP-hard; hence for large graphs G, the right-hand side of (3.4) may be very difficult to

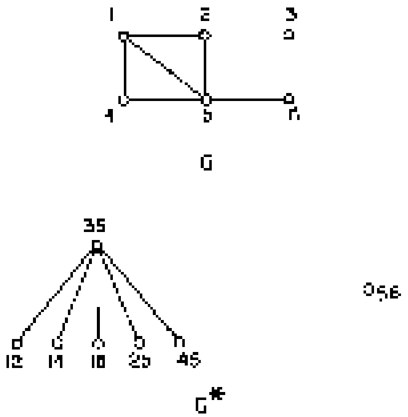


Fig. 5

evaluate. For small graphs, however, (3.4) is quite useful and often precise. In fact, we know of no instance where it holds with the sharp inequality sign.

Problem. Is there a graph G such that $t(G) > \chi(G^*)$?

Remark 2. We shall outline a heuristic for finding a "small" (although not necessarily the smallest) number of threshold graphs $G_i = (V, E_i)$ such that $\bigcup E_i = E$, thereby providing an upper bound on $t(G)$. The heuristic is based on a subroutine for finding a "large" threshold graph $G^0 = (V, E^0)$ with $E^0 \subseteq E$.

The subroutine goes as follows. Given a graph $G = (V, E)$, find a vertex v of the largest degree in G, let S be the set of all the vertices adjacent to v and let $H = (S, T)$ be the subgraph of G induced by S. Applying the subroutine recursively to H, find a "large" threshold graph $H^0 = (S, T^0)$ with $T^0 \subseteq T$. Then define

$$E^0 = T^0 \cup \{uv : u \in S\}$$

and $G^0 = (V, E^0)$.

The heuristic goes as follows. Given a graph $G = (V, E)$, use the subroutine to find a large threshold graph $G^0 = (V, E^0)$ with $E^0 \subseteq E$. Applying the heuristic recursively to the graph $(V, E - E^0)$, find threshold graphs $G_i = (V, E_i)$ with $\bigcup E_i = E - E^0$ and, say, $1 \leq i \leq k$. Then define $G_{k+1} = G^0$.

Clearly, the running time for this heuristic is $O(n^3)$.

4. Pseudothreshold graphs

A switching function $f : \{0, 1\}^n \to \{0, 1\}$ is called pseudothreshold [13] if there are real numbers $a_1, a_2, \ldots, a_n, b$ (not all of them zero), such that, for every zero-one vector $(x_1, x_2, \ldots, x_n)$, we have

$$\sum_{i=1}^{n} a_i x_i < b \implies f(x_1, x_2, \dots, x_n) = 0,$$

$$\sum_{i=1}^{n} a_i x_i > b \implies f(x_1, x_2, \dots, x_n) = 1.$$

By analogy, we shall call a graph *pseudothreshold* if there are real numbers $a(u)$, b ($u \in V$), not all of them zero, such that, for every subset S of V, we have

$$\begin{aligned} \sum_{u \in S} a(u) < b &\implies S \text{ is stable},\\ \sum_{u \in S} a(u) > b &\implies S \text{ is not stable}. \end{aligned} \tag{4.1}$$

In this section, we shall investigate the pseudothreshold graphs. (We do so at the suggestion of the referee of an earlier version of this paper.) In fact, we shall develop an algorithm for deciding whether a graph is pseudothreshold. When G has n vertices, the algorithm terminates within $O(n^4)$ steps; it is not unlikely that this bound may be improved.

We shall begin by making our definition a little easier to work with.

Fact 1. *A graph is pseudothreshold if and only if there are real numbers $a(u)$, b ($u \in V$) such that b is positive and, for every subset S of V, we have* (4.1).

Proof. The "if" part is trivial; in order to prove the "only if" part, we shall consider a pseudothreshold graph $G = (V, E)$. We may assume $E \neq \emptyset$ (otherwise $a(u) = 0$ and $b = 1$ does the job). Since the empty set is stable, (4.1) implies $b \geq 0$. In order to prove $b > 0$, we shall assume $b = 0$ and derive a contradiction. First of all, since every one-point set is stable, we have $a(u) \geq 0$ for every $u \in V$. Secondly, since not every $a(u)$ is zero, there is a vertex w with $a(w) < 0$. Finally, since $E \neq \emptyset$, there are adjacent vertices u and v. Setting $S = \{u, v, w\}$ we contradict (4.1). □

From now on, we shall assume $b > 0$. For every graph $G = (V, E)$ we shall define two subsets P_G, Q_G of V. The set P_G consists of all the vertices u for which there are three other vertices u_1, u_2, u_3 such that

$$uu_1, uu_2, uu_3 \in E; \qquad u_1u_2, u_1u_3, u_2u_3 \notin E.$$

The set Q_G consists of all the vertices v for which there are three other vertices v_1, v_2, v_3 such that

$$vv_1, vv_2, vv_3, v_1v_3 \notin E; \qquad v_1v_2, v_2v_3 \in E.$$

These definitions are illustrated in Fig. 4.

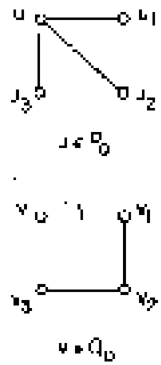


Fig. 4.

Fact 2. *Let $G = (V, E)$ be a pseudothreshold graph. Then*

$$u \in P_0 \Longrightarrow a(u) \geq 2b/3$$

$$v \in Q_0 \Longrightarrow a(v) \leq b/3.$$

Proof. First of all, if $u \in P_0$ then

$$\begin{aligned}
&\qquad\; a(u_1) + a(u_2) + a(u_3) \leq b,\\
&a(u) + a(u_1) \qquad\qquad\qquad\;\; \geq b,\\
&a(u) + \qquad\quad a(u_2) \qquad\quad\; \geq b,\\
&a(u) + \qquad\qquad\qquad a(u_3) \geq b,
\end{aligned}$$

and so $3a(u) \geq 2b$. Secondly, if $v \in Q_0$ then

$$\begin{aligned}
&a(v) + a(v_1) + \qquad\quad a(v_3) \leq b,\\
&a(v) + \qquad\quad a(v_2) \qquad\quad\; \leq b,\\
&\qquad\; a(v_1) + a(v_2) \qquad\quad\; \geq b,\\
&\qquad\qquad\qquad a(v_2) + a(v_3) \geq b,
\end{aligned}$$

and so $3a(v) \leq b$. □

Next, we shall define (by induction on i)

$$P_{i+1} = P_i \cup \{u \in V : uv \in E \text{ for some } v \in Q_i\},$$

$$Q_{i+1} = Q_i \cup \{v \in V : uv \notin E \text{ for some } u \in P_{i+1}\},$$

$$P^* = \bigcup_{i=0}^{\infty} P_i \qquad Q^* = \bigcup_{i=0}^{\infty} Q_i.$$

Fact 3. *If G is a pseudothreshold graph, then $P^* \cap Q^* = \emptyset$.*

Proof. It suffices to prove that

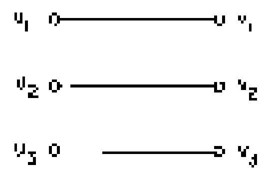
$$u \in P^* \Longrightarrow a(u) \geq 2b/3,$$
$$v \in Q^* \Longrightarrow a(v) \leq b/3;$$

these implications follow easily (by induction on i) from Fact 2. $\square$

From the definition of P^* and Q^*, we readily conclude the following.

Fact 4. *If $P^* \cap Q^* = \emptyset$, then every two vertices in P^* are adjacent and no two vertices in Q^* are adjacent.* $\square$

Our next observation involves the graph $3K_2$ shown in Fig. 5.



$3K_2$

Fig. 5.

Fact 5. *No pseudothreshold graph contains an induced subgraph isomorphic to* $3K_2$.

Proof. Assume the contrary. Then

$$a(u_1) + a(u_2) + a(u_3) \leq b,$$
$$a(v_1) + a(v_2) + a(v_3) \leq b,$$
$$a(u_1) + a(v_1) \geq b,$$
$$a(u_2) + a(v_2) \geq b,$$
$$a(u_3) + a(v_3) \geq b.$$

Trivially, these inequalities are inconsistent with $b > 0$. $\square$

Theorem 4. *For every graph $G = (V, E)$, the following three properties are equivalent:*

(i) *G is pseudothreshold,*

(ii) *$P^* \cap Q^* = \emptyset$ and G has no induced subgraph isomorphic to $3K_2$,*

(iii) *there is a partition of V into pairwise disjoint subsets P, Q and R such that:*

every vertex from P is adjacent to every vertex from $P \cup R$,

no vertex from Q is adjacent to another vertex from $Q \cup R$,

there are no three pairwise nonadjacent vertices in R.

Proof. The implication (i) $\Rightarrow$ (ii) follows from Fact 3 and Fact 5.

To see that (iii) $\Rightarrow$ (i), simply set $b = 2$ and

$$a(u) = \begin{cases} 0 & \text{if } u \in Q, \\ 1 & \text{if } u \in R, \\ 2 & \text{if } u \in P. \end{cases}$$

It remains to be proved that (ii) $\Rightarrow$ (iii). We shall do this by means of a very simple algorithm which terminates in $O(n^2)$ steps either by showing that (ii) does not hold or by constructing the partition described in (iii). The algorithm goes as follows.

First of all, find P^* and Q^*. (This can certainly be done in $O(n^2)$ steps.) Then find out whether $P^* \cap Q^* = \emptyset$. (If not, stop: (ii) does not hold.) Then set $S = V - (P^* \cup Q^*)$; note that by the definition of P^* and Q^*, every vertex from S is adjacent to all the vertices from P^* and to no vertex from Q^*. Let S_0 consist of all the vertices in S which are adjacent to no other vertex in S; define

$$P = P^*, \qquad Q = Q^* \cup S_0, \qquad R = S - S_0.$$

Find out whether there are three pairwise nonadjacent vertices in R. If not, stop: P, Q and R have all the properties described in (iii). If, on the other hand, there are three pairwise nonadjacent vertices $u_1, u_2, u_3 \in R$, then each u_i is adjacent to some $v_i \in R$. Using the fact that $R \cap (P_0 \cup Q_0) = \emptyset$, the reader may now easily verify that the set $\{u_1, u_2, u_3, v_1, v_2, v_3\}$ induces a $3K_2$ in G. (This may be done in the following order. Firstly, the v_i's are distinct. Secondly, each v_i is adjacent to exactly one u_j. Finally, the v_i's are pairwise nonadjacent.) Hence (ii) does not hold. $\square$

Remark. It may be worth pointing out the following corollary of Theorem 1: *If G is pseudothreshold, then one can satisfy (4.1) with $b = 2$ and each $a(u) \in \{0, 1, 2\}$.*

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, (Addison-Wesley, Reading, MA, 1974).

[2] G.H. Bradley, Transformation of integer programs to knapsack problems, *Discrete Math.* 1 (1971) 29–45.

[3] C.K. Chow, Boolean functions realizable with single threshold devices, *Proc. IRE* 49 (1961) 370–371.

[4] V. Chvátal, On Hamilton's ideals, *J. Comb. Theory* 12(B) (1972) 163–168.

[5] V. Chvátal, Tough graphs and Hamiltonian circuits, *Discrete Math.* 5 (1973) 215–228.

[6] S.A. Cook, The complexity of theorem proving procedures, in *Proc. Third Annual ACM Symposium on Theory of Computing* (1971) 151–158.

[7] J. Edmonds, Paths, trees, and flowers, *Canad. J. Math.* 17 (1965) 449–467.

[8] C.C. Elgot, Truth functions realizable by single threshold organs, *IEEE Symposium on Switching Circuit Theory and Logical Design* (1961) 225–245.

[9] P. Erdős, Graph theory and probability, II, *Canad. J. Math.* 13 (1961) 346–352.

[10] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972).

[11] S. Poljak, A note on stable sets and colorings of graphs, *Comm. Math. Univ. Carolinae* 15 (1974) 307–309.

[12] I.G. Rosenberg, Aggregation of equations in integer programming, *Discrete Math.* 10 (1974) 325–341.

[13] T.A. Slivinski, An extension of threshold logic, IEEE TC C-19 (1970) 319–341.

[14] R.O. Winder, Threshold Logic, (Ph.D. thesis), Mathematics Dept., Princeton University, 1962.

Annals of *Discrete Mathematics* 1 (1977) 163–177

ON THE UNCAPACITATED LOCATION PROBLEM*

Gerard CORNUEJOLS,
Department of Operations Research, Cornell University, Ithaca, NY, U.S.A.

Marshall FISHER
Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, PA 19174, U.S.A.

George L. NEMHAUSER
Department of Operations Research, Cornell University, Ithaca, NY, U.S.A.

The problem of optimally locating bank accounts to maximize clearing times is discussed. The importance of this problem depends in part on its mathematical relationship to the well known uncapacitated plant location problem. A Lagrangian dual for obtaining an upper bound and heuristics for obtaining a lower bound on the value of an optimal solution are introduced. The main results are analytical worst-case analyses of these bounds. In particular it is shown that the relative error of the dual bound and a "greedy" heuristic never exceeds $[(K-1)/K]^K < 1/e$ for a problem in which at most K locations are to be chosen. An interchange heuristic is shown to have a worst-case relative error of $(K-1)/(2K-1) < 1/2$. Examples are given showing that all these worst-case bounds are tight. The extreme points of an LP formulation equivalent to the Lagrangian relaxation are also characterized.

The number of days required to clear a check drawn on a bank in city j depends on the city i in which the check is cashed. Thus, to maximize its available funds, a company that pays bills to numerous clients in various locations may find it advantageous to maintain accounts in several strategically located banks. It would then pay bills to clients in city i from a bank in city $j(i)$ that had the largest clearing time. The economic significance to large corporations of locating accounts so that large clearing times can be achieved is discussed in a recent article in *Business-week* [1].

To formalize the problem of selecting an optimal set of account locations, let $I = \{1, \dots, m\}$ be the set of client locations, $J = \{1, \dots, n\}$ the set of potential account locations, d_j the fixed cost of maintaining an account in city j, f_i the fraction of checks paid in city i, ϕ_{ij} the number of days (translated into monetary value) to clear a check issued in city j and cashed in city i, and K the maximum number of accounts that can be maintained. All of this information is assumed to be known and $c_{ij} = f_i\phi_{ij}$ represents the value of paying clients in city i from an account in city j. To simplify the analysis we will also make the realistic assumption that $d_j = 0$ for all j.

* This research was supported by NSF Grants ENG75-00568 and SOC-7402516. Sections 1–4 of this paper include a technical summary of some results given in [2]. Some proofs are omitted and may be obtained in [2].

Let

$$y_j = \begin{cases} 1 & \text{if an account is maintained in city } j, \\ 0 & \text{otherwise,} \end{cases}$$

and x_{ij}, $0 \le x_{ij} \le 1$, be the fraction of customers in city i paid from an account in city j.

The account location problem, which we call (P), can be stated as the integer linear program (IP)

$$z = \max \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} - \sum_{j \in J} d_j y_j \tag{1}$$

$$\text{s.t.} \sum_{j \in J} x_{ij} = 1, \quad i \in I, \tag{2}$$

$$1 \le \sum_{j \in J} y_j \le K, \tag{3}$$

$$x_{ij} \le y_j, \quad i \in I, \quad j \in J, \tag{4}$$

$$y_j \in \{0, 1\}, \quad j \in J, \tag{5}$$

$$x_{ij} \ge 0, \quad i \in I, \quad j \in J. \tag{6}$$

We denote by (LP) the linear program obtained from (IP) by replacing (5) by $0 \le y_j \le 1$, $j \in J$.

The essential variables in (P) are the y_j's since given binary-valued y_j's, say $J^0 = \{j \mid y_j = 1\}$, it is simple to determine an optimal set of x_{ij}'s. Let

$$J^0(i) = \left\{ j \in J^0 \mid c_{ij} = \max_{k \in J^0} c_{ik} \right\}.$$

Then, with respect to J^0, an optimal set of x_{ij}'s is given by $x_{ij} = 1$ for some $j \in J^0(i)$ and $x_{ij} = 0$ otherwise.

There is a vast literature on problems that are mathematically related to problem (P). When $I = J$ are the nodes of a graph, (5) is an equality contraint, and the objective function (1) is replaced by $\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$, the model is known as the K-median problem. When (1) is replaced by

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} d_j y_j,$$

the model is known as the simple or uncapacitated plant or warehouse location problem. [2, 3, 5] contain bibliographies and survey material of applications and methods for this class of problems.

Useful relaxations for a variety of combinatorial problems have been obtained by identifying a set of complicating constraints of the problem, weighting these constraints by multipliers and placing them in the objective function. This dual method is called Lagrangian relaxation. It was first shown to be a very effective computational tool for solving large combinatorial problems by Held and Karp [6, 7] in their work on the traveling salesman problem. Geoffrion [4] has proposed a Lagrangian relaxation for (P) in which one dualizes (P) with respect to the constraints (2). This partial dual is intimately related to the linear program (LP). For example, both problems have the same optimum objective values.

Although most recent work on the class of problems represented generically by problem (P) has been on exact algorithms, there is still a need to study heuristics. Heuristics provide feasible solutions and lower bounds for exact algorithms. Most importantly, however, heuristics appear to be the only reasonable option for solving very large problems. The reason for this pessimistic remark is that problem (P) belongs to the class of problems known to be NP-complete in the sense of Karp [9]. The result that (P) is NP-complete is easily established by reducing the NP-complete node covering problem to (P).

This paper analyzes approximations for problem (P). Our main results are on the quality of solutions obtained from heuristics and upper bounds obtained from linear programming and Lagrangian relaxations.

The paper is organized into five sections. In Section 1 we give a criterion for evaluating heuristics and relaxations. Section 2 describes Geoffrion's Lagrangian relaxation and defines a greedy heuristic. Section 3 contains the derivation of a tight upper bound on the worst performance of the greedy heuristic and the Lagrangian and (LP) relaxations. In Section 4 we formulate and analyze theoretically an interchange heuristic. Although this heuristic is computationally more expensive than the greedy heuristic, we will show that its worst possible performance is inferior to that of the greedy heuristic. Finally, Section 5 provides a characterization of the extreme points of (LP) and discusses implications for deriving cuts.

1. A criterion for measuring the quality of bounds

Let $\mathcal{F}$ be the family of problems generated from problem (P) by considering all positive integer values for m, n, and K, all real $m \times n$ matrices $C = \{c_{ij}\}$ and all real nonnegative n-vectors $d = (d_1, \ldots, d_n)$. As before z denotes the optimal objective value of a particular $P \in \mathcal{F}$. Now let $\bar{z}$ and $\underline{z}$ be upper and lower bounds respectively on z. These bounds may be obtained, for example, from the linear programming relaxation and greedy heuristic, respectively.

When evaluating the quality of a bound — for definiteness say a lower bound — it is not in general meaningful to consider the absolute deviation $z - \underline{z}$, since this deviation is sensitive to scale changes in the data. Thus if there is a (P) that yields a positive absolute deviation, we can construct problems in $\mathcal{F}$ with arbitrarily large deviations.

Relative deviations are more meaningful. However, defining an appropriate measure of relative deviation is subtle. For example, a popular measure of relative deviation for a heuristic in a maximization problem is

$$F = (z - \underline{z})/z = 1 - \underline{z}/z. \tag{7}$$

This measure is appropriate when $z > 0$ in which case $0 \leq F \leq 1$. In a worst case analysis of a particular heuristic one seeks to show that $F \leq \varepsilon < 1$ for all problems within some class. This is equivalent to showing that the ratio $z/\underline{z}$ is bounded by the positive constant $1/(1-\varepsilon)$. Johnson [8] presents a survey of worst case analysis of heuristics for a variety of combinatorial problems in which a measure that is equivalent to (7) is used.

The measure (F) is inadequate for our problem. We cannot require $z > 0$ since a minimization problem such as the simple plant location or K-median problem, when translated into a maximization problem, would generally have $z < 0$. More generally, for our problem, the measure F fails to have the following property that we believe is essential. A modification of the data that adds a constant to the objective value of every feasible solution but leaves the execution of the heuristic unchanged should also leave the error measure unchanged. For example, if a constant δ is added to every element of a row of C in problem (P), then the objective value of each feasible solution is increased by δ but the execution of the greedy heuristic (among others) is unchanged. The measure F is now equal to $(z - \underline{z})/(z + \delta)$ and, provided $z \neq \underline{z}$, it can be made as large (or small) as we like by appropriate choice of δ.

With these considerations in mind, to evaluate lower bounds obtained from a heuristic we use the measure

$$G = (z - \underline{z})/(z - z_R), \tag{8}$$

where z_R is a suitably chosen reference value for (P). Ideally, the reference z_R should equal the minimum objective value of (P) but, in any event, z_R should be a lower bound on this minimum value that is sensitive to significant data changes such as the addition of a constant to every element of a row of C. We may think of $z - z_R$ as the worst absolute deviation that could be achieved by a heuristic. Then G measures the deviation for a particular heuristic relative to the worst possible deviation.

In problem (P) we define

$$z_R = c - KD, \tag{9}$$

where $c_i = \min_{j \in J} c_{ij}$, $c = \sum_{i \in I} c_i$ and $D = \max_{j \in J} d_j$. Thus if $d = 0$ and $c = 0$, $G = F$. Furthermore, if $d = 0$ and $C \geq 0$, we can enforce $G = F$ by adding a fictitious and useless location $(n+1)$, such that $c_{i,n+1} = 0$ for all i.

Our measure for evaluating upper bounds in maximization problems is similar to G. Using the same value for z_R, we define an error measure of an upper bound to be $H = (\bar{z} - z)/(\bar{z} - z_R)$. Note that in H the actual error $(\bar{z} - z)$ is relative to the worst possible error $\bar{z} - z_R$.

We will assume that $\mathscr{P}$ has been restricted to exclude all problems for which $z - z_R = 0$ or $\bar{z} - z_R = 0$. The relations $z_R \leq \underline{z} \leq z \leq \bar{z}$ would make error bound analysis rather pointless in these cases. We note that $0 \le G \le 1$, $0 \le H \le 1$, $G = 0$ if and only if $z_g = z$, and $H = 0$ if and only if $\bar{z} = z$.

2. A Lagrangian relaxation and the greedy heuristic

Let x be the matrix whose elements are x_{ij}, $i \in I$, $j \in J$, $y = (y_1, \ldots, y_n)$,

$$S = \{x, y \mid x, y \text{ satisfies constraints (3), (4), (5) and (6)}\},$$

and $u = (u_1, \ldots, u_m)$ be multipliers for the constraints (2). A Lagrangian problem for (P) is given by

$$z_D(u) = \max_{x,y \in S} \left\{ \sum_{i \in I} \sum_{j \in J} (c_{ij} - u_i) x_{ij} - \sum_{j \in J} d_j y_j + \sum_{i \in I} u_i \right\}$$

$$= \max_{x,y \in S} \left\{ \sum_{j \in J} \left[\sum_{i \in I} (c_{ij} - u_i) x_{ij} - d_j y_j \right] + \sum_{i \in I} u_i \right\}$$

and the corresponding Lagrangian dual by

$$z_D = \min_u z_D(u).$$

It is well-known that $z_D \ge z$. Furthermore, since the matrix defined by the constraints (2) and (4) is totally unimodular, it follows from a theorem of Geoffrion [4] that z_D is equal to the optimum value of the linear programming relaxation (LP).

Define

$$\rho_j(u) = \sum_{i \in I} \max(0, c_{ij} - u_i) - d_j.$$

Observe that $\rho_j(u)$ is the potential contribution of location j to $z_D(u)$, since in an optimal solution city i will be assigned to a selected location j if and only if $c_{ij} - u_i > 0$. Thus to determine $z_D(u)$ for fixed u we define $J^+(u) = \{j \in J \mid \rho_j(u) > 0\}$ and set $J(u) = J^+(u)$ if $1 \le |J^+(u)| \le K$. Otherwise let $J(u)$ be an index set corresponding to the K largest $\rho_j(u)$ if $|J^+(u)| > K$ or the single largest $\rho_j(u)$ if $|J^+(u)| = 0$. We then have

Proposition 1. *An optimal solution to*

$$\max_{x,y \in S} \sum_{j \in J} \left[\sum_{i \in I} (c_{ij} - u_i) x_{ij} - d_j y_j \right]$$

is given by

$$y_j = \begin{cases} 1 & \text{if } j \in J(u), \\ 0 & \text{otherwise;} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if } y_j = 1 \quad \text{and} \quad c_{ij} - u_i > 0, \\ 0 & \text{otherwise.} \end{cases}$$

As a consequence of Proposition 1 we have that

$$z_D(u) = \sum_{j \in J(u)} \rho_j(u) + \sum_{i \in I} u_i. \tag{10}$$

In studying the Lagrangian relaxation we observed that if $J^* \subset J$ represents a set of selected locations and $u_i = \max_{j \in J^*} c_{ij}$ then $\rho_j(u)$, $j \notin J^*$, represents the improvement in the objective function if we augment J^* by j. This observation leads quite naturally to the conception of the following "greedy" heuristic for (P). This heuristic is suggested by Spielberg [10]. The greedy heuristic first chooses a location to solve (P) for $K = 1$ and then proceeds recursively. Suppose $k < K$ locations have been selected. If there exists an unselected location that improves the value of the objective function, choose one that yields the maximum improvement; otherwise stop.

The greedy heuristic

Step 1: Let $k = 1$, $J^0 = \emptyset$ and $u_i^1 = \min_{j \in J} c_{ij}$, $i \in I$.

Step 2: Let $\rho_j(u^k) = \sum_{i \in I} \max(0, c_{ij} - u_i^k) - d_j$, $j \notin J^{k-1}$. If $\rho_j(u^k) \leq 0$ for all $j \notin J^{k-1}$ and $|J^{k-1}| \geq 1$ set $k = k - 1$ and go to Step 5. Otherwise, go to Step 3.

Step 3: Find $j_k \notin J^{k-1}$ such that $\rho_{j_k}(u^k) = \max_{j \notin J^{k-1}} \rho_j(u^k)$. Set $J^k = J^{k-1} \cup \{j_k\}$. If $|J^k| = K$ go to Step 5, otherwise go to Step 4.

Step 4: Set $k = k + 1$. For $i \in I$ set

$$u_i^k = \max_{j \in J^{k-1}} c_{ij} = u_i^{k-1} + \max(0, c_{ij_{k-1}} - u_i^{k-1}).$$

Go to Step 2.

Step 5: Stop; the greedy solution is given by $y_j = 1$, $j \in J^k$, $y_j = 0$, otherwise. We have $|J^k| = k$ and the value of the greedy solution is

$$z_g = \sum_{i \in I} u_i^1 + \sum_{t=1}^{k} \rho_{j_t}(u^t).$$

The following example illustrates the greedy heuristic with $d = 0$, $K = 2$ and

$$C = \begin{pmatrix} 0 & 11 & 6 & 9 \\ 7 & 0 & 8 & 2 \\ 7 & 3 & 0 & 3 \\ 10 & 9 & 4 & 0 \end{pmatrix}.$$

We initialize with $J^0 = \emptyset$ and $u^1 = (0, 0, 0, 0)$. Then $\rho_1(u^1) = 24$, $\rho_2(u^1) = 23$, $\rho_3(u^1) = 18$, $\rho_4(u^1) = 14$, $j_1 = 1$ and $J^1 = \{1\}$. We set $u^2 = (0, 7, 7, 10)$ and obtain $\rho_2(u^2) = 11$, $\rho_3(u^2) = 7$ and $\rho_4(u^2) = 9$. Thus $j_2 = 2$, $J^2 = \{1, 2\}$, and $z_g = 35$. We also note that $z_D(u^2) = 35 + 1 + 0 = 36$ so that $35 \leq z \leq 36$.

3. Analysis of the greedy heuristic and Lagrangian dual

In this section we show that

$$(z_D - z_G)/(z_D - z_R) \leq 1/e \quad \text{for all} \quad P \in \mathcal{P}. \tag{11}$$

Since $z_R \leq z_G \leq z \leq z_D$, (11) implies that

$$G_P = (z - z_G)/(z - z_R) \leq 1/e, \tag{12}$$

$$H_P = (z_D - z)/(z_D - z_R) \leq 1/e. \tag{13}$$

We will present examples to show that these are the best possible bounds; that is $\sup_{P \in \mathcal{P}} G_P = \sup_{P \in \mathcal{P}} H_P = 1/e$. Furthermore, since $z_D = z_{LP}$, the optimal value of the linear programming relaxation (LP), we obtain the result that $(z_{LP} - z)/(z_{LP} - z_R) \leq 1/e$. Let $\mathcal{P}_K$ be the subfamily of $\mathcal{P}$ in which at most K locations may be selected.

Lemma 1. *For all $P \in \mathcal{P}_K$*

$$(z_D - z_G)/(z_D - z_R) \leq [(K-1)/K]^K \leq 1/e.$$

Proof. If $K = 1$ or $\rho_1(u^1) \leq 0$, the theorem is clearly true since $z_G = z_D(u^1)$. Otherwise let k be the number of locations selected by the greedy heuristic and $\bar{k}$ the number of times Step 2 of the greedy heuristic is executed. If $k = K$ then $\bar{k} = k$; otherwise $\bar{k} = k + 1$. In either case $\bar{k} \geq 2$. Let $\alpha = (K-1)/K$ and, for notational simplicity, $\rho_j = \rho_j(u^j)$, $j = 1, \ldots, \bar{k} - 1$ and $\rho_{\bar{k}} = \rho_{\bar{k}}(u^{\bar{k}})$ if $\bar{k} = k$, $\rho_{\bar{k}} = 0$ if $\bar{k} = k + 1$. The statement of the lemma is equivalent to

$$(1 - \alpha^K) z_D + \alpha^K z_R \leq z_G. \tag{14}$$

For $s = 1, \ldots, \bar{k}$, $\sum_{i \in I} u_i^s = c + \sum_{j=1}^{s-1} (\rho_j - d_j)$ and $K\rho_s$ is nonnegative and at least as large as the K largest $\rho_j(u^s)$, $j \notin J^s$. Using these facts and $D \geq d_j$, $j \in J$, we obtain from (10)

$$z_D \leq c - \sum_{j=1}^{s-1} \rho_j + K\rho_s + (s-1)D, \quad s = 1, \ldots, \bar{k}. \tag{15}$$

We will establish the lemma by showing that (14) holds when z_D is replaced by the minimum of the bounds given in (15).

Let $a_s = -\sum_{j=1}^{s-1} \rho_j + K\rho_s + (s-1)D$ so that (15) becomes $z_D \leq c + a_s$, $s = 1, \ldots, \bar{k}$. Substituting in (14) $z_R = c - KD$, $z_G = c + \sum_{j=1}^{k} \rho_j$ and the bounds for z_D, we must show that

$$(1 - \alpha^K) \min_s (c + a_s) + \alpha^K (c - KD) \leq c + \sum_{j=1}^{k} \rho_j$$

or (cancelling terms in c)

$$(1 - \alpha^K) \min_s a_s - K\alpha^K D \leq \sum_{j=1}^{k} \rho_j. \tag{16}$$

We establish (16) by assuming

$$(1-\alpha^K)a_s - K\alpha^K D \geq \sum_{j=1}^{s} \rho_j, \quad s = 1,\dots,\bar{k}-1 \tag{17}$$

and showing that (17) implies

$$(1-\alpha^K)a_{\bar k} - K\alpha^K D \leq \sum_{j=1}^{\bar k} \rho_j. \tag{18}$$

Substituting for $a_{\bar k}$ in (18) and simplifying yields

$$(1-\alpha^K)\sum_{j=1}^{\bar k-1} \rho_j + (1-\alpha^K)K\rho_{\bar k} - \sum_{j=1}^{\bar k} \rho_j \leq [(1-\bar k)(1-\alpha^K) + K\alpha^K]D. \tag{19}$$

Multiply inequality s of (17) by $\alpha^{\bar k-1-s}$ and sum for $s = 1,\dots,\bar k-1$ to obtain

$$\begin{aligned}(1-\alpha^K)\sum_{s=1}^{\bar k-1}\alpha^{\bar k-1-s}\left[\sum_{j=1}^{s}\rho_j + (s-1)D + K\rho_s\right] - K(1-\alpha^{\bar k-1})K\alpha^K D \geq \\ \geq K(1-\alpha^{\bar k-1})\sum_{j=1}^{\bar k}\rho_j.\end{aligned} \tag{20}$$

where we have used the fact that $\sum_{s=1}^{\bar k-1}\alpha^{\bar k-1-s} = (1-\alpha^{\bar k-1})/(1-\alpha) = K(1-\alpha^{\bar k-1})$. It can be shown that (20) can be simplified to

$$(1-\alpha^K)\sum_{j=1}^{\bar k-1}\rho_j - (1-\alpha^{\bar k-1})\sum_{j=1}^{\bar k}\rho_j \geq [K-\bar k+1-\alpha^K+\bar k\alpha^K - K\alpha^{\bar k-1}]D. \tag{21}$$

We now consider two cases to show that (21) implies (19).

Case (a). $[k = K = \bar k]$: Here (19) reduces to

$$\alpha^K\sum_{j=1}^{K-1}\rho_j - (K-1)(1-\alpha^{K-1})\rho_K \leq [(1-K)(1-\alpha^K) + K\alpha^K]D \tag{22}$$

and (21) reduces to

$$(\alpha^{K-1}-\alpha^K)\sum_{j=1}^{K-1}\rho_j - (1-\alpha^{K-1})\rho_K \geq [1-\alpha^K + K(\alpha^K-\alpha^{K-1})]D. \tag{23}$$

Multiplying (23) by $-\alpha^K/(\alpha^{K-1}-\alpha^K) = 1-K < 0$ implies (22), which completes the proof of case (a).

Case (b). $[k < K, \bar k = k+1]$: Here (19) reduces to

$$-\alpha^K\sum_{j=1}^{k}\rho_j + (1-\alpha^K)K\rho_{k+1} \leq [-k(1-\alpha^K) + K\alpha^K]D \tag{24}$$

and (21) reduces to

$$(\alpha^k-\alpha^K)\sum_{j=1}^{k}\rho_j \geq [K-k-k\alpha^K - K\alpha^k]D. \tag{25}$$

Multiplying (25) by $-\alpha^K/(\alpha^k-\alpha^K) < 0$ yields

$$\alpha^K \sum_{i=1}^{k} p_i \le -\alpha^K[K - k + k\alpha^K - K\alpha^k]D/(\alpha^k - \alpha^K).$$

Since, in this case, we have $p_{k+1} = 0$ and $D \ge 0$, (24) will be implied by the above inequality if

$$-\alpha^k[K - k - k\alpha^K - K\alpha^k]/(\alpha^k - \alpha^K) \ge -k(1-\alpha^K) + K\alpha^K. \tag{26}$$

The inequality (26) simplifies to

$$k \ge K\alpha^{K-k}. \tag{27}$$

We prove (27) by induction. For $k = K-1$ we have $K\alpha^{K-k} = K-1$ so that (27) is an equality for all K. Now assume that (27) is true for k and consider $k-1$. We have

$$K\alpha^{K-k+1} = K\alpha^{K-k} - K\alpha^{K-k}(1-\alpha) = K\alpha^{K-k} - \alpha^{K-k}$$
$$\ge K\alpha^{K-k} - 1 \ge k-1,$$

where the last inequality is implied by the induction hypothesis. $\square$

As immediate consequences of Lemma 1 and the relations $z_R \le z_L \le z \le z_D$ we have the following two Theorems.

Theorem 1. *For all $P \in \mathcal{P}_K$, $G_s \ge [(K-1)/K]^K$.*

Theorem 2. *For all $P \in \mathcal{P}_K$, $H_D \ge [(K-1)/K]^K$.*

We now give two families of problems in $\mathcal{P}_K$, $K = 2, 3, \ldots$ which show that $[(K-1)/K]^K$ is a tight bound for G_s and H_D respectively. ($K = 1$ is trivial). Either family also implies the tightness of the bound in Lemma 1.

Theorem 3. *Let $P \in \mathcal{P}_K$, $K = 2, 3, \ldots$, be defined by $m = K(K-1)$, $n = 2K-1$, $d = 0$ and C^K where for $j = 1, \ldots, K-1$*

$$c_{ij}^K = \begin{cases} (K-1)K^{K-j}\alpha^{j-1}[\alpha - (K-1)/K], & i = (j-1)K+1, \ldots, jK, \\ 0, & \text{otherwise}, \end{cases}$$

and for $j = K, \ldots, 2K-1$

$$c_{ij}^K = \begin{cases} K^{K-1}, & i = l + j - (l-2)K,\ l = 1, \ldots, K-1 \\ 0, & \text{otherwise}. \end{cases}$$

Then $G_s = [(K-1)/K]^K$, $K = 2, 3, \ldots$.

Proof. See [7].

Theorem 4. *Let* $P_r \in \mathcal{P}_K$, $r = 2, 3, \ldots$, $K = 2, 3, \ldots$, *be defined by* $d = 0$, $n = Kr$, $m = \binom{n}{r}$ *and the* 0–1 *matrix* $C^{(r)}$, *where the rows of* $C^{(r)}$ *consist of all* 0–1 n-*vectors with precisely* r *positive elements. Then*

$$z_R = 0, \qquad z_G = m = \binom{n}{r}, \qquad z = \sum_{s=1}^{K} \binom{n-s}{r-1}$$

and, for each K, *as* r *approaches infinity* H_G *approaches* $[(K-1)/K]^K$.

Proof. See [2].

4. The interchange heuristic

In this section we do a worst-case analysis of an interchange heuristic. It will be convenient throughout this section to treat the subfamily of $\mathcal{P}$ with $d = 0$. Since in this subfamily (P) always has an optimal solution that uses K locations, the interchange heuristic will take a particularly simple form. The heuristic is initialized with an arbitrary set $J^0 \subseteq J$ of cardinality K. With respect to J^0 optimal values for the x_{ij} are chosen in the obvious manner mentioned in the introduction. We then determine if the solution can be improved by augmenting J^0 by a location not in J^0 and deleting from J^0 one of its present members. The procedure continues in this way until no such interchange yields an improvement. In the worst-case analysis it is not necessary to specify details on how the particular entering-leaving pair is selected such as first improvement vs. maximum improvement.

Theorems 5 and 6 characterize the relative error of this heuristic. For problem (P) let z_I be the value of the solution produced by the interchange heuristic, $G_I = (z - z_I)/(z - z_R)$, and $\mathcal{P}_K$ the subfamily of $\mathcal{P}$ in which $d = 0$ and at most K locations are to be selected.

Theorem 5. *For all* $P \in \mathcal{P}_K$, $G_I \leq (K-1)/(2K-1)$.

Proof. See [2].

Theorem 6. *Let* $P \in \mathcal{P}_K$, $K = 1, 2, \ldots$, *be defined by* $m = 2K - 1$, $n = 2K$ *and*

$$C^K = \begin{bmatrix} 1 & & & & & & & & 1 \\ & 1 & & & & 0 & & & 1 \\ & & \ddots & & & & & & \vdots \\ & & & \ddots & & & & & 1 \\ & & & & \ddots & & & & 0 \\ & 0 & & & & \ddots & & & 0 \\ & & & & & & \ddots & & \vdots \\ & & & & & & & 1 & 0 \end{bmatrix}$$

(The first $2K-1$ columns of C^K are unit vectors and the last column has K one's.) Then $G_I = (K-1)/(2K-1)$, $K = 1, 2, \dots$.

Proof. The first K columns are an interchange solution since if any column j, $K+1 \le j \le 2K$, is interchanged for one of the first K columns, the increase in the objective function is 0. This gives $z_I = K$. The last K columns are an optimal set, so $z = 2K-1$. Since $z_R = 0$, $G_I = (2K-1-K)/(2K-1) = (K-1)/(2K-1)$. $\square$

Since the interchange heuristic can begin with an arbitrary set of locations of cardinality K, we might choose an initial solution by applying the greedy heuristic. We will call the method that begins with the greedy solution and then applies the interchange heuristic the "greedy-interchange" heuristic. Let z_{gI} be the value of the solution produced by the greedy interchange heuristic and $G_{gI} = (z - z_{gI})/(z - z_R)$. The family of worst-case problems used in Theorem 3 show that we can have $z_{gI} > z_g$. However, there is a family of problems for which $G_g = [(K-1)/K]^K$ and no improvements can be made by applying the interchange heuristic. In particular we have

Theorem 7. *Let $P \in \mathscr{P}_K$, $K = 2, 3, \dots$, be defined by $m = K^K$, $n = 2K$ and the matrix C^K, where for $1 \le j \le K$*

$$c_{ij}^K = \begin{cases} (K-1)^{j-1}K^{K-j}, & i = (j-1)K+1, \dots, jK \\ 0, & \text{otherwise} \end{cases}$$

$$c_{i,K+j}^K = \begin{cases} K^{K-1}, & i = lK + j,\ l = 0, \dots, K-1 \\ 0, & \text{otherwise.} \end{cases}$$

Then $G_g = G_{gI} = [(K-1)/K]^K$, $K = 2, 3, \dots$.

Proof. See [2].

5. The extreme points of (LP)

In Section 3 we studied the relationship between (P) and (LP) in terms of their objective values. These problems may also be compared by studying the extreme points of their underlying polyhedra. It is easy to show that any solution to the (IP) formulation of (P) is also an extreme point of (LP). In this section we complete the description of the LP polyhedron by characterizing the fractional extreme points of (LP).

For a given non-integer solution (x, y) of (LP) let $J_1 = \{j \in J \mid 0 < y_j < 1\}$ and $I_1 = \{i \in I \mid x_{ij} = 0$ or y_j for all j and x_{ij} non-integer for some $j\}$.

Let

$$a_{ij} = \begin{cases} 1 & \text{if } x_{ij} > 0, \\ 0 & \text{if } x_{ij} = 0, \end{cases}$$

and denote by A the $|I_2| \times |J_f|$ matrix whose elements are a_{ij} for $i \in I_2$ and $j \in J_f$.

Theorem 8. *The non-integer solution (x, y) of (LP) is an extreme point of the LP polyhedron if and only if*

(i) $y_j = \max_i x_{ij}$ *for all* $j \in J_f$,

(ii) *for each* $i \in I$, *there is at most one* j *with* $0 < x_{ij} < y_j$,

(iii) *the rank of* A *equals* $|J_f|$.

Proof. The proof will use the well-known fact that (x, y) is extreme if and only if each pair of solutions (x^1, y^1) and (x^2, y^2) to (LP) that satisfy $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$, $y = \frac{1}{2}y^1 + \frac{1}{2}y^2$ also satisfy $x^1 = x^2$ and $y^1 = y^2$.

We first show that (x, y) is not extreme if (i) or (ii) are violated. If (i) is violated there exists a k such that $\max_i x_{ik} < y_k < 1$. Let $\varepsilon = \min(y_k - \max_i x_{ik}, 1 - y_k)$ and set

$$y_k^1 = y_k + \varepsilon, \qquad y_k^2 = y_k - \varepsilon, \qquad y_j^1 = y_j^2 = y_j, \quad j \neq k,$$

$$x_{ij}^1 = x_{ij}^2 = x_{ij} \quad \text{for all } i \text{ and } j.$$

Then since (x^1, y^1) and (x^2, y^2) are feasible in (LP) and satisfy $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$, $y = \frac{1}{2}y^1 + \frac{1}{2}y^2$ the fact that $y^1 \neq y^2$ implies (x, y) is not extreme. A similar argument may be used in the case where (ii) is violated if we let k, j_1 and j_2 denote indices satisfying $0 < x_{kj_1} < y_{j_1}$, $0 < x_{kj_2} < y_{j_2}$ and set

$$\varepsilon = \min\{x_{kj_1}, y_{j_1} - x_{kj_1}, x_{kj_2}, y_{j_2} - x_{kj_2}\} > 0,$$

$$x_{kj_1}^1 = x_{kj_1} + \varepsilon, \qquad x_{kj_2}^1 = x_{kj_2} - \varepsilon, \qquad x_{kj_1}^2 = x_{kj_1} - \varepsilon,$$

$$x_{kj_2}^2 = x_{kj_2} + \varepsilon, \qquad x_{ij}^1 = x_{ij}^2 = x_{ij} \quad \text{for all other } ij \text{ and } y_j^1 = y_j^2 = y_j \text{ for all } j.$$

We now represent general (x^1, y^1) and (x^2, y^2) as $x_{ij}^1 = x_{ij} + \delta_{ij}$, $x_{ij}^2 = x_{ij} - \delta_{ij}$, $y_j^1 = y_j + \delta_j$ and $y_j^2 = y_j - \delta_j$ for $i \in I$, $j \in J$ where δ_{ij} and δ_j are selected so that (x^1, y^1) and (x^2, y^2) are feasible in (LP). We will complete the proof by showing that when (i) and (ii) are satisfied, any such δ_{ij} and δ_j satisfy $\delta_{ij} = \delta_j = 0$ if and only if the rank of A is $|J_f|$. Let

$$J_0 = \{j \in J : y_j = 0\},$$

$$J_1 = \{j \in J : y_j = 1\},$$

$$I_1 = \{i \in I : x_{ij} \text{ integer for } j \in J\} \text{ and}$$

$$I_2 = \{i \in I : 0 < x_{ij} < y_j \text{ for precisely one } j \in J\}.$$

Note that J_0, J_1, J_f and I_1, I_2, partition J and I.

It is immediate from the upper and lower limits on y_j and x_{ij} imposed by the constraints of (LP) that $\delta_j = 0$, $j \in J_0 \cup J_2$, $\delta_{ij} = 0$, $i \in I$, $j \in J_0$, and $i \in I_2$, $j \in J$. Also, by the definition of I_1 and J_2, $x_{ij} = 0$ and hence $\delta_{ij} = 0$ for $i \in I_1$, $j \in J_2$.

This leaves δ_j, $j \in J_1$ and δ_{ij}, $i \in I_1$, $j \in J_1$ and $i \in I_3$, $j \in J_1 \cup J_2$ undetermined. For $i \in I_1$, $j \in J_1$, $\delta_{ij} = 0$ if $x_{ij} = 0$, and if $x_{ij} > 0$, then

$$\delta_{ij} = \delta_j, \quad i \in I_1, \quad j \in J_1, \tag{28}$$

because $x^1_{ij} = x_{ij} + \delta_{ij} \leq y_j + \delta_j$ and $x_{ij} = y_j$ implies $\delta_{ij} \leq \delta_j$ while $x^2_{ij} = x_{ij} - \delta_{ij} \leq y_j - \delta_j$ and $x_{ij} = y_j$ implies $\delta_{ij} \geq \delta_j$. We may then use constraint (2) of (LP) to impose

$$\sum_{j \in J_1} a_{ij}\delta_j = 0, \quad i \in I_1. \tag{29}$$

For $i \in I_3$, $j \in J_1 \cup J_2$, let $j(i)$ denote the unique index for which $0 < x_{ij(i)} < y_{j(i)}$. The feasibility requirements of (LP) imply $\delta_{ij} = 0$ if $j \in J_1$ and $x_{ij} = 0$ or $j \in J_2 - \{j(i)\}$; and if $x_{ij} > 0$

$$\delta_{ij} = \delta_j, \quad i \in I_3, \quad j \in J_1 - \{j(i)\}. \tag{30}$$

Constraints (2) and (6) will be satisfied if

$$\sum_{j \in J_1 - \{j(i)\}} a_{ij}\delta_j = -\delta_{ij(i)}, \quad i \in I_3, \tag{31}$$

$$|\delta_{ij(i)}| \leq \min(x_{ij(i)}, y_{j(i)} - x_{ij(i)}), \quad i \in I_3. \tag{32}$$

Feasible values for those δ_j and δ_{ij} that are not immediately equal to zero are now completely determined by (28)–(32). If the rank of the coefficient matrix A of (29) is $|J_1|$, then $\delta_j = 0$ for $j \in J_1$ is the unique solution of (29). Equations (28) and (30)–(31) then imply that the remaining $\delta_{ij} = 0$ so that x, y is extreme. If the rank of A is less than $|J_1|$, then let $\bar{\delta}_j$, $j \in J_1$ denote a nonzero solution to (29). Since $\alpha\bar{\delta}_j$, $j \in J_1$ satisfies (29) for any α, we may determine values for δ_{ij} using (28), (30), and (31) and select α sufficiently small that (32) is satisfied. This implies that x, y is not extreme. $\square$

If the rank of A equals $|J_1|$, then A contains a $|J_1| \times |J_1|$ nonsingular submatrix. Let B denote such a submatrix and e a $|J_1|$-component vector of ones. The fractional part of x, y may be completely determined from the unique solution to $Bz = e$ by setting

$$y_j = z_j, \; j \in J_1, \tag{33}$$

$$x_{ij} = z_j, \qquad x_{ij} > 0, \qquad i \in I_1, \; j \in J_1 \text{ or } i \in I_3, \; j \in J_1 - \{j(i)\}, \tag{34}$$

$$x_{ij(i)} = 1 - \sum_{j \in J_1 - \{j(i)\}} x_{ij}, \quad i \in I_3. \tag{35}$$

Intuitively, B contains the "fractional information" of (x, y) and should be useful in determining a cut which removes (x, y). An example of such a relationship is afforded by a class of extreme points that are generated from the solution to $Bz = e$ when B is a generalized cycle matrix. Let $C^{tk} = \{c_{ij}^{tk}\}$ denote the $k \times k$ matrix whose rows are 0–1 vectors in which t contiguous ones are successively moved one position to the right.

For example

$$C^{34} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

C^{tk} is nonsingular if (and only if) t and k are relatively prime, in which case C^{tk} may be used to generate an extreme point of (LP) by selecting $I_1 \subset I$ and $J_1 \subset J$ with $|I_1| = |J_1| = k$ and solving the system

$$\sum_{j \in J_1} c_{ij}^{tk} z_j = 1, \quad i \in I$$

with coefficient matrix C^{tk}. An extreme point is obtained by using the unique solution $z_j = 1/t$, $j \in J_1$ of this system to determine fractional y_j and x_{ij} from (33)–(35) and selecting any feasible integer values for the remaining y_j and x_{ij}. It is interesting that a cut which removes this extreme point may also be determined from the matrix C^{tk}. This cut is

$$\sum_{i \in I_1} \sum_{j \in J_1} c_{ij}^{tk} x_{ij} - \sum_{j \in J_1} y_j \leq k - \lceil k/t \rceil$$

where $\lceil k/t \rceil$ denotes the least integer greater than or equal to k/t. This inequality is valid for any k and t and is a cut, that is it removes part of the (LP) feasible region if k/t is not integer. However, in the process of removing these fractional extreme points, it is certainly possible to create new ones.

References

[1] Businessweek, Making millions by stretching the float, 45–90, (November 23, 1974) 88–90.

[2] G. Cornuejols, M.L. Fisher and G.L. Nemhauser, An analysis of heuristics and relaxations for the uncapacitated plant location problem, Management Sci., 23 (8) (April 1977).

[3] R.L. Francis and J.M. Goldstein, Location theory: A selective bibliography, Operations Res. 22 (1974) 400–410.

[4] A.M. Geoffrion, Lagrangian relaxation for integer programming, *Math. Programming Study* 2 (1974) 82–114.

[5] A.M. Geoffrion, A guide to computer-assisted methods for distribution systems planning, Western Man. Sci. Center, Paper No. 216, UCLA, 1974.

[6] M. Held and R.M. Karp, The traveling salesman problem and minimum spanning trees, *Operations Res.* 18 (1970) 1138–1162.
[7] M. Held and R.M. Karp, The traveling salesman problem and minimum spanning trees: Part II, *Math. Programming* 1 (1971) 6–25.
[8] D.S. Johnson, Approximation algorithms for combinatorial problems, *J. of Computer and Systems Sciences* 9 (1974) 256–278.
[9] R.M. Karp, On the computational complexity of combinatorial problems, *Networks* 5 (1975) 45–68.
[10] K. Spielberg, Algorithms for the simple plant location problem with some side conditions, *Operations Res.* 17 (1969) 85–111.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 179–184

SOME COLORING TECHNIQUES

D. de WERRA

Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Two types of colorings for graphs and hypergraphs are considered here: good and equitable colorings. By using several techniques (namely partial colorings and node splitting) we study some classes of graphs (and hypergraphs) which have k-colorings of the above types for given values of k. Some new results on edge colorings are obtained by combining these coloring techniques.

1. Introduction

In this paper the terminology of Berge [2] will be used. By *k-coloring* of a hypergraph $H = (X, \mathscr{E})$ we simply mean a partition of its node set X into k subsets $U_1, \ldots, U_k$. For multigraphs $G = (X, E)$ we will deal only with edge colorings; so a *k-coloring of G* will be a partition of its edge set E into k subsets $U_1, \ldots, U_k$; it is in fact a k-coloring of the dual hypergraph G^* of G.

A *usual* coloring of H is a coloring where not all nodes in the same edge E' have the same color (i.e. are in the same U_i) if $|E'| \geq 2$; for a graph G a coloring of G is usual if no two adjacent edges are of the same color.

Several extensions of usual colorings have been proposed; the most interesting so far seem to be the *equitable colorings* [7] and the *good colorings* [3].

For a k-coloring $(U_1, \ldots, U_k)$ of H we denote by $u_i(E')$ the cardinality of $U_i \cap E'$ ($E' \in \mathscr{E}$ is an edge of H); in the same way, if $(U_1, \ldots, U_k)$ is a k-coloring of G, $u_i(x)$ will be the number of edges in U_i which are adjacent to node x.

$(U_1, \ldots, U_k)$ is an *equitable* k-coloring if

$e(E') = \max_{i,j \leq k} |u_i(E') - u_j(E')| \leq 1$ for each edge E' of H, or

$e(x) = \max_{i,j \leq k} |u_i(x) - u_j(x)| \leq 1$ for each node x of G.

It is a *good* k-coloring if

$k(E') = |\{i \mid u_i(E') \geq 1\}| = \min(k, |E'|)$ for each E' of H, or

$k(x) = |\{i \mid u_i(x) \geq 1\}| = \min(k, d(x))$ for each x in G.

In this paper we first intend to describe some classes of graphs which have k-colorings of the above types for all $k \geq s$ where s is a fixed number.

Besides if we are given a graph G and a positive integer k, G may not have a k-coloring of the above types; however it may have a *deficient* k-coloring (i.e. a k-coloring which is good or equitable except possibly for a certain subset S of nodes). We will also try to characterize subsets S of this kind.

In order to obtain the above mentioned results we shall apply two coloring techniques: partial coloring and node splitting. These methods are described in the next sections.

2. Partial colorings

In the remainder of the paper a generalized k-coloring will be an equitable or a good k-coloring. A *partial* coloring of a hypergraph H (resp. of a graph G) is a coloring of a sub-hypergraph of H (resp. of a partial graph of G).

For some coloring theorems constructive proofs based on the idea of partial colorings (or more precisely recolorings) have been given. These theorems have the following form. Let $S(p)$ be any sufficient condition for the existence of a generalized p-coloring.

Theorem 2.1. *Let H be a hypergraph such that any subhypergraph of H satisfies $S(p)$; then, for each $k \geq p$, H has a generalized k-coloring.*

A possible proof technique may be the following [3, 7]: starting from any k-coloring $(U_1, \dots, U_k)$ of H one determines a subhypergraph H' generated by $\bigcup_{i=1}^{p} U_i$ for which $(U_1, \dots, U_p)$ is not a generalized p-coloring. Since H' satisfies $S(p)$, there exists a partial p-coloring $(U_1', \dots, U_p')$ of H' which is a generalized p-coloring. One verifies separately for good and for equitable colorings that $(U_1', \dots, U_p', U_{p+1}, \dots, U_k)$ is a k-coloring of H which is better than $(U_1, \dots, U_k)$ in the sense that for at least one edge some measure of quality of the coloring has increased and for no edge has this measure of quality decreased. By repeated use of this procedure, one finally gets a generalized k-coloring as required.

Remark 2.1. Notice that this type of proof would not be applicable to other types of colorings such as r-bounded colorings for instance (i.e. colorings satisfying

$$r(E') = \max_{i,j \leq k} [u_i(E') - u_j(E')] \leq r \quad \text{where } r \geq 2).$$

Application 1 [7]. *A unimodular hypergraph has an equitable k-coloring for each $k \geq 2$.*

This follows from the characterization of totally unimodular matrices given by A. Ghouila-Houri [6].

The measure of quality for edge E' is here the number of pairs of colors i, j for which $u_i(E') - u_j(E')$ does not exceed a given value.

Application 2 [5]. *A balanced hypergraph has a good k-coloring for each $k \geq 2$.*

This is a direct consequence of the fact that any subhypergraph of a balanced hypergraph has a good bicoloring [2, p. 452]. In this case the measure of quality for edge E' is $k(E')$ i.e. the number of colors appearing in edge E'. When restricted to the case of graphs (or more generally of multigraphs) Theorem 2.1 becomes

Theorem 2.1A. *Let G be a multigraph such that any partial multigraph satisfies $S(p)$; then, for each $k \geq p$, G has a generalized k-coloring.*

Application 1 [7]. *A bipartite multigraph has an equitable k-coloring (and hence a good k-coloring) for each $k \geq 2$.*

Here $S(2)$ is the property of containing no odd cycles.

Now this conclusion may be generalized as follows:

Application 2 [8]. *Let G be a multigraph such that in each odd cycle C there exist two consecutive nodes which are not joined by an odd chain in $G - C$ to any node of C. Then G has an equitable k-coloring for each $k \geq 3$.*

(This result is obtained by showing that the above property can be taken for $S(3)$).

Application 3 [9]. *If in any partial multigraph of G the edges may be oriented in such a way that for each node x, either $d^+(x) \equiv 0 \pmod p$ or $d^-(x) \equiv 0 \pmod p$, then G has an equitable k-coloring for each $k \geq p$.*

Remark 2.2. Similar conditions $S(p)$ could be given for the existence of good p-colorings in applications 2 and 3.

Notice finally that application 2 could be formulated in another way. Given a multigraph G, we might say that an odd cycle C which does not have the above described property is a *strong* odd cycle. (Thus in a strong odd cycle C among any two consecutive nodes of C there is at least one which is joined by an odd chain in $G - C$ to a node of C.) Then if k is given and if S is a subset of nodes which meet all strong odd cycles, $G = (X, E)$ has a k-coloring which is equitable for all nodes $x \in X - S$ (i.e. $e(x) \leq 1$) but for nodes $x \in S$ we may have $e(x) \leq 2$. (The construction of k-colorings with $e(x) \leq 2$ is always possible as shown in [7]).

In the next section we shall try to describe other sets S where we may not have $e(x) \leq 1$ but possibly $e(x) \leq 2$.

3. Node splitting

If G is a simple graph, it is known that its chromatic index $q(G)$ satisfies $d \leq q(G) \leq d + 1$ where d is the maximum degree in G. This is Vizing's theorem;

an elegant proof has been given by J.C. Fournier [5]. Beineke and Wilson [1] as well as Fiorini and Wilson [4] say that a simple graph G is *of class* 1 if $q(G) = d$ and *of class* 2 otherwise. We will extend this definition to multigraphs.

Before proceeding further we need to introduce the idea of node splitting. Given a multigraph G and a positive integer k, we may apply a k-*splitting operation* to the nodes of G; this will result in a multigraph G_k obtained as follows: for each node x with degree $d(x) > k$, the $d(x)$ edges adjacent to x are numbered arbitrarily; x is split into $\lceil d(x)/k \rceil$ nodes $x', x'', \ldots, x^{(p)}$ ($\lceil t \rceil$ denotes the smallest integer not less than t). x' is adjacent to the first k edges, x'' to the next k edges and so on (only the last node $x^{(p)}$ may be adjacent to less than k edges). Clearly G_k will have maximum degree k.

We will say that a property P of G is s-*stable* if any G_k obtained by a k-splitting of G with $k \geq s$ also has property P. For instance if P is the property of having no odd cycles, then P is 2-stable. But if P is the property that no connected component of G is an odd cycle, then it is not 2-stable.

Theorem 3.1. *Let G be a multigraph and $s \geq 2$ an integer; if for any $k \geq s$ G as well as any G_k obtained by k-splitting are of class 1, then G has an equitable k-coloring.*

Proof. Any usual k-coloring of G_k gives obviously an equitable k-coloring of G.

At this point we might derive the same applications as in the previous section by choosing in each case a suitable property P. (We would have $s = 2$ in the first application and $s = 3$ in the second one.) We will however concentrate on other properties.

Applying a coloring procedure which is not a partial recoloring in the sense defined above, J.C. Fournier has obtained the following result [5]:

Let G be a simple graph with maximum degree d; if there is no cycle meeting only nodes of degree d, then G is of class 1.

Using Fournier's theorem we get

Application 1. *Let G be a simple graph and $h \geq 2$ an integer such that each cycle contains at least one node with degree $< h$; then, for each $k \geq h$, G has an equitable k-coloring.*

If h is the maximum degree d, this is just the result of Fournier; and if $h < d$ any G_k is of class 1 from Fournier's theorem (the property P of having in each cycle at least one node with degree $< h$ is h-stable).

This result may be extended to some classes of multigraphs since k-splitting operations may sometimes transform multigraphs into simple graphs. Here $m(x, y)$

is the multiplicity of the pair x, y of nodes, i.e. the number of parallel edges joining nodes x and y.

Application 2. *Let G be a multigraph such that in each cycle there exists at least one node with degree $< h$. Let p be the largest integer such that $m(x, y) \leq \lceil d(x)/p \rceil \lceil d(y)/p \rceil$ for each pair of nodes x, y. Then if $p \geq h$, G has an equitable k-coloring for each k with $h \leq k \leq p$.*

Proof. If $k \leq p$ one may construct a G_k which is a simple graph. Furthermore in each cycle of G_k there will be at least one node with degree $< k$ (since $k \geq h$). Hence Fournier's theorem may be applied.

Illustration. The multigraph G consisting of 3 nodes a, b, c and 5 edges (a, b), (a, c), $(b, c)_1$, $(b, c)_2$, $(b, c)_3$ is such that $p = 3$ since

$$m(b, c) = 3 \leq \lceil 4/3 \rceil \lceil 4/3 \rceil = \lceil d(b)/p \rceil \lceil d(c)/p \rceil.$$

We may take $k = 3$. G_3 obtained by the 3-splitting operation has nodes a, b, b', c, c' and edges (a, b), (a, c), (b, c), (b', c) and (b, c'). It is a simple graph of class 1. One sees that G neither has an equitable 2-coloring nor an equitable 4-coloring.

Remark 3.1. Analogous results for good colorings could be derived by devising an adequate k-splitting operation: each node x with degree $d(x) > k$ is split into one node x' of degree k and one or more other nodes of arbitrary degree not exceeding k.

For hypergraphs node splitting operations would become edge splitting operations. However this procedure cannot be used in the same way as for graphs: while any k-splitting applied to a bipartite multigraph still gives a bipartite multigraph, any edge k-splitting operation acting on a balanced hypergraph may not produce a balanced hypergraph.

As a conclusion we may combine several coloring techniques such as partial coloring, node splitting and Fournier's coloring procedure. This gives the following:

Theorem 3.2. *Let G be a simple graph and let $h \geq 2$ be such that each cycle where all nodes have degrees at least h contains at least one node which does not belong to any strong odd cycle. Then G has an equitable k-coloring for each $k \geq h$.*

Proof. This result is obtained by first constructing G_k with k-splitting operations and then determining any k-coloring. Then the recoloring procedure described by Fournier [5] is applied until either the quality of the coloring is improved (i.e. as previously the measure of quality has been increased for at least one node) or a node x with degree k has been reached and x does not belong to any strong odd cycle.

In this case the partial coloring procedure (used for application 2 of theorem 2.7.A [8]) may be applied. This will also improve the quality of the coloring. By iterating this procedure one eventually gets an equitable k-coloring.

Theorem 3.2 could also be formulated in an alternative way.

Let G be a simple graph and h a positive integer; let F be the family of all cycles which contain only nodes having degree at least h and belonging to some strong odd cycle. If S is a subset of nodes meeting all cycles in F, then for each $k \geq h$ *G has a* k*-coloring satisfying* $c(x) \leq 1$ *for any node* $x \in X - S$ *and* $c(x) \leq 2$ *for any* $x \in S$.

References

[1] L.W. Beineke, R.J. Wilson, On the edge chromatic number of a graph, Discrete Math. 5 (1973) 15–20.

[2] C. Berge, Graphs and Hypergraphs (North Holland, Amsterdam, 1973).

[3] C. Berge, Notes sur les bonnes colorations d'un hypergraphe, Cahiers du C.E.R.O., 15 (1973) 219–223.

[4] S. Fiorini, R.J. Wilson, On the chromatic index of a graph, I, Cahiers du C.E.R.O., 15 (1973) 253–262.

[5] J.C. Fournier, Colorations des arêtes d'un graphe, Cahiers du C.E.R.O. 15 (1973) 311–314.

[6] A. Ghouila-Houri, Caractérisation des matrices totalement unimodulaires, C.R. Acad. Sci., Paris, 254 (1962) 1192.

[7] D. de Werra, Equitable colorations of graphs, R.I.R.O., R-3, 1971, pp. 3–8.

[8] D. de Werra, An extension of bipartite multigraphs, Discrete Math. 14 (1976) 133–138.

[9] D. de Werra, "How to color a graph", in: B. Roy, ed. Combinatorial Programming: Methods and Applications (Reidel, Dordrecht, 1975) 305–325.

Annals of Discrete Mathematics 1 (1977) 185–204
© North-Holland Publishing Company

A MIN-MAX RELATION FOR SUBMODULAR FUNCTIONS ON GRAPHS

Jack EDMONDS and Rick GILES*
CORE, Université Catholique de Louvain, B-3030 Heverlee, Belgium

1. Introduction

(1.0) We prove here a new combinatorial min-max equality which unifies and extends results including the matroid intersection theorem [4] and the theorem of Lucchesi and Younger on the minimum number of edges which meet every directed cut in a graph [14]. Like matroid intersection theory and optimum matching theory [15], the subject is developed as statements on the existence of integer-valued optima to certain large combinatorially described linear programs.

The method of proof used here generalizes the method used in [6] to prove the polymatroid intersection theorem and the method used in [13] to prove the Lucchesi–Younger Theorem including an idea which Lovász attributes to Neil Robertson.

We are especially grateful to Ellis Johnson for his help on this work.

The present section states the main theorem. Sections 2–6 discuss several cases of it. Sections 7, 8 and 9 prove it. The results in Sections 7 and 8 are also of interest in themselves. Special cases of Section 7 appear in a number of places. Section 8 extends the idea of Robertson and a main idea of [2]. Section 10 proves a consequence of the main theorem, and also places the theorem in a setting which we call "box total dual integrality".

(1.1) Let $G = (V, E)$ be a directed graph with node-set V and edge-set E, where each $e \in E$ has tail $t(e) \in V$ and head $h(e) \in V$.

(1.2) For $S \subseteq V$, let

$$\delta(S) = \{e \in E : t(e) \in S, h(e) \notin S\}.$$

For $S \subseteq V$, let $\bar{S} = V - S$. For $v \in V$, let $\bar{v} = V - \{v\}$ and let v be used sometimes for $\{v\}$.

(1.3) A family F of subsets of V is called a *crossing family* on V if

* The research of each author is partially supported by a grant from the National Research Council of Canada.

$$S \cap T \in F, \qquad S \cup T \in F,$$

for any two sets $S \in F$ and $T \in F$ such that

$$S \cap T \neq \emptyset, \qquad S \cup T \neq V.$$

(1.4) For any family F of subsets of V, a real-valued function $f(S)$, $S \in F$, is called *submodular on F* if

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T)$$

for all $S, T \in F$ such that $S \cap T$, $S \cup T \in F$.

For any vector $x = (x_e : e \in E) \in \mathbf{R}^E$, and any $H \subseteq E$, let

$$x(H) = \sum (x_e : e \in H).$$

For any given graph $G = (V, E)$, crossing family F on V, submodular function f on F, and vectors $a, d, c \in (\mathbf{R} \cup \{\pm\infty\})^E$, consider the linear program,

(1.5) maximize cx,

(1.6a) where $d \leq x \leq a$,

(1.6b) $\forall S \in F, x(\delta(S)) - x(\delta(\bar{S})) \leq f(S)$.

(1.7) For $y = (y_S : S \in F) \in \mathbf{R}^F$, let

$$yf = \sum (y_S f(S) : S \in F),$$

$$F(y, e) = \sum (y_S : S \in F, e \in \delta(S)) - \sum (y_S : S \in F, e \in \delta(\bar{S})).$$

The linear programming dual of (1.5) is

(1.8) minimize $yf + za - wd$

where $y \in \mathbf{R}^F$, $z \in \mathbf{R}^E$, and $w \in \mathbf{R}^E$

(1.9) satisfy $y \geq 0$, $z \geq 0$, $w \geq 0$,

$\forall e \in E, z_e - w_e + F(y, e) = c_e$.

The l.p. duality theorem says that:

(1.10) The maximum in (1.5) equals the minimum in (1.8), assuming either of these optima exists.

(1.11) **Theorem.** *If c is integer-valued, and linear program (1.8) has an optimum solution, then it has an integer-valued optimum solution. Hence, if c is integer-valued, (1.10) holds even when restricted to integer-valued solutions $[y, z, w]$ of (1.9).*

(1.12) **Theorem.** *If a, d, and f are integer-valued, and linear program (1.5) has an optimum solution, then it has an integer-valued optimum solution. Hence, if a, d, and* *f are integer-valued,* (1.10) *holds even when restricted to integer-valued solutions x of* (1.6).

Using a simple fact of linear programming, Theorem (1.12) is immediately equivalent to:

(1.13) *If a, d, and f are integer-valued, then every non-empty face of the polyhedron P of the system (1.6) contains an integer point. In particular, if P has a vertex, then every vertex of P is an integer point.*

2. Network Flows

(2.0) *Let* $G = (V, E)$ be a graph; let $d, a \in (\mathbf{R} \cup \{\pm\infty\})^E$, and let $r, q \in (\mathbf{R} \cup \{\pm\infty\})^V$. A *feasible flow in network G* in the classical sense of [10] is a vector $x \in \mathbf{R}^E$ which satisfies

(2.1) $d \le x \le a$,

$$r_v \le x(\delta(v)) - x(\delta(\bar v)) \le q_v \quad \text{for all } v \in V.$$

Let $F_1 = \{\{v\}: v \in V\}$ and $F_2 = \{\bar v: v \in V\}$. Let $F = F_1 \cup F_2$. Let $f(\{v\}) = q_v$ and $f(V - v) = -r_v$.

Clearly, F is a crossing family, f is submodular on F, and (1.6) for this case is (2.1). Theorems (1.11) and (1.12) for this case are well known.

3. Polymatroids

(3.0) For a matroid M defined on the set E, the *rank function* of M is $f(S) = |J|$ for any maximal $J \subseteq S$ such that J is independent in M. (For example, where E is the set of indices of the columns of a matrix A, and where $J \subseteq E$ is independent in M when the set of columns indexed by J is linearly independent.)

(3.1) The rank function $f(S)$, $S \subseteq E$, of a matroid on E is submodular; it is non-decreasing: $A \subseteq B \subseteq E$ implies $f(A) \le f(B)$; $f(\emptyset) = 0$; and for each $e \in E$, $f(\{e\}) \le 1$. Such an f determines its matroid, say M, by the fact that J is independent in M iff $|J| = f(J)$.

(3.2) Let f be any submodular function of all subsets of E. Let $a \in (\mathbf{R} \cup \{\pm\infty\})^E$. The polyhedron,

$$P_a = \{x \in \mathbf{R}^E : 0 \le x \le a;\ x(S) \le f(S),\ \forall S \subseteq E\},$$

known as a *polymatroid*, is much like the family of independent sets of a matroid, (3.3) Furthermore, Theorems (1.11)–(1.13) hold where the linear programs (1.5)–(1.6) and (1.8)–(1.9) are replaced by

(3.4) maximize $\{cx : x \in P_0\}$

and the dual of (3.4).

(3.5) This follows immediately from (1.11)–(1.13) by letting E of (3.2) be the edge-set of a graph $G = (V, E)$ such that the heads and the tails of the members of E are all different;

(3.6) letting the F of (1.5) be

$$F = \{\{t(e) : e \in S\} : S \subseteq E\};$$

and letting the f of (1.5) be

$$f(\{t(e) : e \in S\}) = f(S), \quad \text{for } S \subseteq E, \text{ as in (3.2)}.$$

Theorem (3.3) is especially simple when

(3.7) the vector a is all infinite, and when

(3.8) $f(S)$, $S \subseteq E$, is a non-negative, non-decreasing submodular function.

The linear program (3.4) becomes

(3.9) maximize $cx = \sum (c_e x_e : e \in E)$,

where $\forall e \in E,\ x_e \geq 0$,

$$\forall S \subseteq E, \sum (x_e : e \in S) \leq f(S).$$

The dual l.p. is

(3.10) minimize $yf = \sum (f(S) \cdot y(S) : S \subseteq E)$,

where $\forall S \subseteq E,\ y(S) \geq 0$,

$$\forall e \in E, \sum (y(S) : e \in S \subseteq E) \geq c_e.$$

The so-called "Greedy Algorithm Theorem" says that:

(3.11) In the case of (3.7)–(3.9), and where the vector c is arranged so that

$$c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_k} > 0 \geq c_{e_{k+1}} \geq \cdots \geq c_{e_{|E|}},$$

the following vectors $x^0 = (x^0_{e_i} : i = 1, \ldots, |E|)$ and $y^0 = (y^0(S) : S \subseteq E)$ are optimum solutions, respectively, of (3.9) and (3.10).

(3.12) Let $S_i = \{e(1), e(2), \dots, e(i)\}$.

(3.13) Let $x^0_{e(1)} = f(S_1)$, $x^0_{e(i)} = f(S_i) - f(S_{i-1})$ for $i = 2, \dots, k$; and $x^0_{e(i)} = 0$ for $i = k+1, \dots, |E|$.

(3.14) Let $y^0(S_i) = c_{e(i)} - c_{e(i+1)}$ for $i = 1, \dots, k-1$; $y^0(S_k) = c_{e(k)}$; and $y^0(S) = 0$ for other $S \subseteq E$.

That these are optimum solutions of (3.9) and (3.10) follows, using the weak l.p. duality theorem, by showing that $cx^0 = y^0 f$, that y^0 is feasible for (3.10), and that x^0 is feasible for (3.9).

It follows from the greedy algorithm theorem that:

(3.15) The vertices of $P_f = \{x \geq 0 : x(S) \leq f(S), \forall S \subseteq E\}$ are the vectors of the form x^0, as defined in (3.13).

(3.16) In particular, where f is the rank function of a matroid, the vectors x^0 are the (incidence) vectors of the independent sets of M. That is, $x^0_e = 1$ for $e \in J$ and $x^0_e = 0$ for $e \in E - J$, where $J \subseteq E$ is an independent set of M.

(3.17) An interesting way to get a function f of the form (3.8) is to take a non-negative linear combination of the rank functions of various matroids on E.

(3.18) Another way to get a very particular kind of f of the form (3.8) is to let $f(S) = g(|S|)$, $S \subseteq E$, where g is a non-negative, non-decreasing, concave function. That is, for $i = 0, 1, 2, \dots, |E|$,

$$g(i) = g(0) + h(1) + h(2) + \cdots + h(i),$$

where $g(0) \geq 0$ and $h(1) \geq h(2) \geq \cdots \geq h(|E|) \geq 0$.

(3.19) In particular, for the f of (3.18), where $g(0) = 0$, we have immediately from (3.15), that a vector is a vertex of P_f iff its components are any arrangement of $h(1), h(2), \dots, h(k)$, and $|E| - k$ zeroes for some k.

(3.20) Hence, the face $P_f \cap \{x : x(E) = f(E)\}$ of the P_f of (3.19) is the convex hull of the vectors which are the various permutations of the numbers $h(1), \dots, h(|E|)$.

The greedy algorithm theorem, as presented here, and some other theory of polymatroids, first appeared in [6]. Further, and better, treatments are [9] and [12].

Balas [1] recently presented a different derivation of a linear system defining the convex hull of the vectors of all permutations of the numbers $1, 2, \dots, |E|$. We much appreciate the thoughtfulness which Chvátal devoted to bringing together Balas' work and ours.

4. Polymatroid intersection

(4.0) Let f_1 and f_2 be any two submodular functions of all subsets of E. Let $a \in (\mathbf{R} \cup \{+\infty\})^E$.

(4.1) For $i = 1, 2$, let

$$P_i = \{x \in \mathbf{R}^E : 0 \leq x \leq a;\ x(S) \leq f_i(S),\ \forall S \subseteq E\}.$$

As in the last section, each P_i is a polymatroid.

(4.2) The polyhedron $P_1 \cap P_2$ is not generally a polymatroid.

(4.3) Nevertheless we do have the "Polymatroid Intersection Theorem" which is (1.10)–(1.13) where linear programs (1.5) and (1.8) are replaced by

(4.4) $\max\{cx : x \in P_1 \cap P_2\}$,

and its l.p. dual.

(4.5) Where we take the intersection of three polymatroids, $P_1 \cap P_2 \cap P_3$, in place of two, the (1.11)–(1.13) part of (4.3) is generally not true. Of course, the (1.10) part still holds, it being merely an instance of the l.p. duality theorem.

(4.6) We get (4.3) as a special case of (1.10)–(1.13) by letting the E of (4.0) be the edge-set of the same graph $G = (V, E)$ as in (3.5), that is, such that each $e \in E$ and its end-nodes comprise a separate component of G; letting $d = 0$;

(4.7) letting $F = \{t(S) : S \subseteq E\} \cup \{\overline{h(S)} : S \subseteq E\}$

where $t(S) = \{t(e) : e \in S\}$ and

$$\overline{h(S)} = V - \{h(e) : e \in S\} = \{t(e) : e \in E\} \cup \{h(e) : e \notin S\};$$

(4.8) letting $f(t(S)) = \min[f_1(S), k]$ for $S \subseteq E$,

$$f(\overline{h(S)}) = \min[f_2(S), k] \quad \text{for } S \subseteq E,$$

where $k = \min[f_1(E), f_2(E)]$.

It is straightforward to verify that F is a crossing family of V, that f is a submodular function of F, and that for this F, f and d, the system (1.6) is equivalent to the system

(4.9) $0 \leq x \leq a$,

$$\forall S \subseteq E:\ x(S) \leq f_1(S),\quad x(S) \leq f_2(S).$$

(4.10) Where f_1 and f_2 are the rank functions of any two matroids on E, say M_1 and M_2, the polymatroid intersection theorem becomes the "matroid intersection theorem":

The (1.13) part immediately implies that:

(4.11) Where P_i is the polyhedron of matroid M_i on set E, $i = 1, 2$, the vertices of $P_1 \cap P_2$ are precisely the vectors of subsets of E which are independent in both M_1 and M_2, that is, they are precisely the points which are vertices of both P_1 and P_2!

Likewise the (1.12) aspect of the matroid intersection theorem (when $a = \infty$) gives us that:

(4.12) The maximum weight, $\sum (c_e : e \in J)$, of a set $J \subseteq E$ which is independent in both matroids, M_1 and M_2, equals

(4.13) $\min \sum (f_1(S) \cdot y_1(S) + f_2(S) \cdot y_2(S) : S \subseteq E)$

where

$$\forall S \subseteq E,\ y_1(S) \geq 0,\ y_2(S) \geq 0,$$

$$\forall e \in E,\ \sum (y_1(S) + y_2(S) : e \in S \subseteq E) \geq c_e.$$

And the (1.12) part gives us that:

(4.14) If c is integer-valued then the $y_i(S)$, $S \subseteq E$, $i = 1, 2$, of (4.13) may be restricted to integers.

For the case where c is all ones, equation (4.12)–(4.14) reduces to:

(4.15) $\max\{|J| : J$ independent in M_1 and $M_2\}$

$$= \min\{f_1(S) + f_2(E - S) : S \subseteq E\}.$$

(4.16) The polymatroid intersection theorem, where the f_i are non-decreasing and without the constraint $x \leq a$, and the matroid instances of it, first appear in [6]. Algorithmic proofs of matroid instances were obtained and published earlier, [5, 8]. The theorem, with the constraint $x \leq a$ and without the restriction on f_i, as well as the main generalization (1.10)–(1.13) being presented here, first appears in [12].

5. Directed cut k-packings

Let $G = (V, E)$ be an acyclic graph and let

(5.0) $D(G) = \{S : \emptyset \neq S \neq V, \delta(\bar{S}) = \emptyset\}$

(5.1) Clearly, $D(G)$ is a crossing family on V. A set of edges of the form $\delta(S)$ for some $S \in D(G)$ is called a *directed cut* of G.

(5.2) For a given integer-valued function $f(S)$, $S \in D(G)$, a set $H \subset E$ such that

$$\forall S \in D(G), \quad |H \cap \delta(S)| \leq f(S),$$

is called a *directed cut f-packing* of G. The incidence vectors of the directed cut f-packing of G are precisely the integer solutions of the system

(5.3) $\forall e \in E, \quad 0 \leq x_e \leq 1,$

$$\forall S \in D(G), x(\delta(S)) \leq f(S)$$

(5.4) When $f(S)$ is submodular, in particular when $f(S)$ is a constant integer k, system (5.3) is of the form (1.6) and so theorems (1.10)–(1.13) apply.

For a constant k, directed cut k-packings are easily treated without the present theory.

The theorem of Dilworth on the maximum number of incomparable elements in a partial order immediately implies that:

(5.5) A subset H of the edges of an acyclic graph G is contained in the edge-set of as few as k directed paths in G iff $|T| \leq k$ for any $T \subseteq H$ such that

(5.6) no directed path of G contains more than one member of T.

It can be shown that

(5.7) a set $T \subseteq H$ has property (5.5) iff T is contained in some member of $D(G)$.

Hence, we have that

(5.8) a set $H \subseteq E$ is a directed cut k-packing in G, for constant integer k, if and only if H is contained in the edge-set of some k or fewer directed paths in G.

(5.9) Corollary. *A set $H \subset E$ is a directed cut k-packing in G, for constant integer k, if and only if H can be partitioned into some k or fewer 1-packings of the directed cuts in G.*

It follows directly from (5.8) that:

(5.10) For a given acyclic graph $G = (V, E)$, a given integer k, and given edge-weighting $c = (c_e: e \in E)$, the maximum weight directed cut k-packings of

G' can be realized as the optimum integer flows of the optimum network flow problem described in Section 2.

(5.11) where the G of Section 2 is the G' of (5.10), with the same edge-weighting, together with, for each $e \in E'$, k extra edges in parallel with e and each having weight of zero; also let G have a new node S, k new zero-weighted edges going from S to each $v \in V'$, a new node t, and k new zero-weighted edges going from each $v \in V'$ to t. Let d be all zeroes, a be all ones, $r_s = q_s = k$, $r_t = q_t = -k$, and $r_v = q_v = 0$ for $v \in V'$.

For the case $k = 1$, and r all ones, the subject of this section is treated by Vidyasankar and Younger [16].

6. Directed cut k-coverings

Let G and $D(G)$ be as in Section 5.

(6.0) Where $g(S)$ is a non-negative integer valued function of $S \in D(G)$, a set $C \subseteq E$ such that $|C \cap \delta(S)| \geq g(S)$ for every $S \in D(G)$ is called a directed cut g-covering of G.

The incidence vectors of the directed cut g-coverings of G are precisely the integer solutions of the system

(6.1) $\forall e \in E,\quad 0 \leq x_e \leq 1,$

$$-x(\delta(\bar{S})) \leq f(S) = -g(\bar{S})$$

for every $S \in F = \{S \subseteq V : \bar{S} \in D(G)\}$.

(6.2) A function $g(S)$ is called *supermodular* when $-g(S)$ is submodular.

(6.3) When $g(S)$ is supermodular, in particular a constant k, the system (6.1) is of the form (1.6) and so Theorems (1.10)–(1.13) apply. The integer min-max relation of (1.10)–(1.12) becomes:

(6.4) Where $g(S)$, $S \in D(G)$, is any integer supermodular function such that

$$0 \leq g(S) \leq |\delta(S)| \quad \text{for every } S \in D(G),$$

where c_e, $e \in E$, are integers, and C is a direct-cut g-covering of G, we have

(6.5) $\min \sum \{c_e : e \in C\}$

$$(6.6)\qquad = \max \sum (y_S \cdot g(S) : S \in D(G)) - \sum (z_e : e \in E)$$

$$(6.7)\qquad = \max \Big\{ \sum (y_S \cdot g(S) : S \in D(G)) - \sum \Big[\max \Big\{0, -c_e + \sum (y_S : e \in \delta(S))\Big\} \Big] : e \in E \Big\}$$

over integers $y_S \geq 0$ and $z_e \geq 0$ such that,

$$\forall e \in E, \qquad z_e \geq \sum (y_S : e \in \delta(S)) - c_e$$

In particular, where the c_e are all ones, formula (6.5)-(6.7) becomes

(6.8) Theorem. *The minimum cardinality of a directed-cut g-covering of G equals the maximum over all*

(6.9) $Y \subseteq D(G)$ of

$$|\cup(\delta(S): S \in Y)| + \sum (g(S) - |\delta(S)| : S \in Y).$$

Where $g(S)$ is all ones, (6.8) implies the theorem of Lucchesi and Younger [24] that:

(6.10) The minimum cardinality of a 1-covering of the directed cuts of G equals the maximum cardinality of a family of mutually disjoint directed cuts of G.

(6.11) A graph $G = (V, E)$ is called *strongly connected* when, for every $u, v \in V$, there is a directed path in G from u to v. A connected graph G is strongly connected if and only if every $e \in E$ is contained in a directed polygon (directed cycle) in G.

(6.12) It is easy to show that $C \subseteq E$ is a 1-covering of the directed cuts of a connected graph G if and only if the graph obtained from G by "shrinking" the members of C is strongly connected — equivalently, if and only if the graph obtained from G by adjoining to G, for each $e \in C$, an edge e' such that $h(e') = t(e)$ and $t(e') = h(e)$, is strongly connected.

We hope to be able to prove the following conjecture:

(6.13) For any constant integer $k > 0$, $C \subseteq E$ is a k-covering of the directed cuts of $G = (V, E)$ if and only if C can be partitioned into k 1-coverings of the directed cuts of G.

(6.14) The function $|\delta(S)|$, $S \in D(G)$, is modular — that is, it is both submodular and supermodular. Hence, though we derived directed-cut f-packings, for submodular f, and directed-cut g-coverings, for supermodular g, as different special cases of a more general system, in fact the two are equivalent: H is an f-packing for G if and only if $E - H$ is a g-covering for G, where

$$g(S) = |\delta(S)| - f(S), \quad S \subset D(G).$$

7. Total dual integrality

(7.0) We say that a system, $Ax \leq b$, of linear inequalities in x, with rational A and b, is *totally dual integral* when the dual of the linear program $\max\{cx : Ax \leq b\}$ has an integer-valued optimum solution for every integer-valued c such that it has an optimum solution. We say that a polyhedron is *totally dual integral* if it is the solution-set of a totally dual integral system.

(7.1) Theorem. *If a polyhedron P is the solution-set of a totally dual integral system which has integer right-hand sides, then every non-empty face of P contains an integer point — in particular, any vertex of P is an integer point.*

Or, stated another way:

(7.1′) Theorem. *For any finite linear system, $Ax \leq b$, having rational coefficients, if $\min\{yb : y \geq 0, yA = c\}$ is an integer for any integer-valued c such that the minimum exists, then for any c such that $\max\{cx : Ax \leq b\}$ exists there is an integer-valued optimum x.*

(7.2) Using Theorem (7.1) we can conclude (1.12) immediately from (1.11).

To prove (7.1) we use the following lemma which we presume to be classical.

(7.3) A finite system of linear equations, $A^0x = b^0$, having rational coefficients, has no integer-valued solution x if and only if there is a vector π such that πA^0 is integer-valued, and πb^0 is not an integer.

Proof of (7.1). Assume the hypothesis of (7.1) for the system $Ax \leq b$. Let $P = \{x : Ax \leq b\}$. By the l.p. duality theorem we have immediately that

(7.4) $\max\{cx : x \in P\}$ is an integer for any integer-valued c such that the maximum exists.

A face of P is any subset of the form $P^0 = \{x \in P : A^0x = b^0\}$ where $A^0x \leq b^0$ is a subsystem of $Ax \leq b$. It is easy to show that

(7.5) if P^0 is a minimal non-empty face of P, then $P^0 = \{x : A^0x = b^0\}$. By the complementary slackness theorem of linear programming, for any c such that

$\max\{cx : x \in P\}$ exists, the maximum is achieved over all members of some non-empty face of P, and hence over all members of some minimal non-empty face of P. Thus it suffices to show that every minimal non-empty face of P, say $P^0 = \{x : A^0x = b^0\}$, has an integer-valued member. Suppose not. Then, by (7.3), let π be such that πA^0 is an integer-valued vector and πb^0 is a non-integer.

Any $c = \lambda A^0$, for a vector $\lambda \geq 0$, is such that cx is maximized over P by any member of P^0, since for $x \in P^0$ we have $cx = \lambda A^0 x = \lambda b^0$, and for $x \in P$ we have $cx = \lambda A x \leq \lambda b^0$.

Choose $\lambda \geq 0$ such that $\lambda + \pi \geq 0$ and such that $c^1 = \lambda A^0$ is integer-valued. Then $c^0 = (\lambda + \pi)A^0$ is integer-valued. By (7.4), for $i = 0, 1$, $d^i = \max\{c^i x : x \in P\}$ is an integer. By (7.5), for $i = 0, 1$, we have $c^i x = d^i$ for every x satisfying $A^0 x = b^0$. Hence, $d^0 - d^1 = c^0 x - c^1 x = \pi A^0 x = \pi b^0$ is an integer. Contradiction. □

8. Tree representation of cross-free families

(8.0) Two sets $S, T \subseteq V$ are said to *cross* if $S \cap T \neq \emptyset$, $S \cup T \neq V$, $S \not\subseteq T$, and $T \not\subseteq S$. A family F of subsets of V is called a *cross-free family on* V if no two members of F cross.

(8.1) A tree T, with node-set $V(T)$, and with directed edge-set $E(T)$, together with a function l from a set V to $V(T)$, is called a *V-labelled tree* T.

(8.2) For any V-labelled tree T, we have a family $\{S_i : i \in E(T)\}$ of subsets of V determined as follows: for each $i \in E(T)$, there is a unique $T(i) \subseteq V(T)$ such that, with respect to graph T, $\delta(T(i)) = \{i\}$, $\delta(\overline{T(i)}) = \emptyset$; $T(i)$ is the set of nodes a (including the node $a = t(i)$) such that the unique path in T from a to $t(i)$ does not contain i. We let

$$S_i = \{v \in V : l(v) \in T(i)\}.$$

(8.3) **Theorem.** *A family F on set V is a cross-free family if and only if*

(8.4) *there exists a V-labelled tree T such that*

$$F = \{S_i : i \in E(T)\}.$$

Proof. It is easy to check that (8.4) implies F is cross-free.

If F consists of just one set S, then let T consist of a single edge i and, for each $v \in V$, let $l(v) = t(i)$ if $v \in S$, and $l(v) = h(i)$ if $v \notin S$. Clearly T and l are a V-labelled tree T satisfying (8.4).

If F' is a cross-free family on V such that $|F'| \geq 2$, choose some $S \in F'$ and let $F = F' - \{S\}$. Assume, by induction on $|F'|$, that we have a V-labelled tree T with labelling function l which satisfies (8.4).

(8.5) Let $l(S) = \{l(v): v \in S\}$, let $\bar{S} = V - S$, let $\bar{T}(i) = V(T) - T(i)$, etc.

Let T_1 and T_2 be the unique minimal subtrees of T such that

$$l(S) \subseteq V(T_1), \qquad l(\bar{S}) \subseteq V(T_2).$$

If $|V(T_1) \cap V(T_2)| \geq 2$ then there is an edge $i \in E(T_1) \cap E(T_2)$. However, $i \in E(T_1)$ implies $T(i) \cap l(S) \neq \emptyset$ and $\bar{T}(i) \cap l(S) \neq \emptyset$, and $i \in E(T_2)$ implies $T(i) \cap l(\bar{S}) \neq \emptyset$ and $\bar{T}(i) \cap l(\bar{S}) \neq \emptyset$. Hence, S_i and S cross, which contradicts F' being cross free.

Therefore, we have $|V(T_1) \cap V(T_2)| \leq 1$, and so we can extend T_1 and T_2 respectively to subtrees T_1' and T_2' of T such that, for some node $u \in V(T)$, we have

$$V(T_1') \cap V(T_2') = \{u\}, \qquad V(T_1') \cup V(T_2') = V(T),$$

$$l(S) \subseteq V(T_1'), \qquad l(\bar{S}) \subseteq V(T_2').$$

Let T' be the tree, and let l' be the V-labelling of T', defined as follows:

$$V(T') = (V(T) - \{u\}) \cup \{u_1, u_2\} \quad \text{where } u_1, u_2 \notin V(T).$$

$$E(T') = E(T) \cup \{e'\} \quad \text{where } e' \notin E(T).$$

For each $e \in E(T')$, the head $h'(e)$ of e in T' is the same as the head $h(e)$ of e in T and the tail $t'(e)$ of e in T' is the same as the tail $t(e)$ of e in T, except

$$t'(e') = u_1; \qquad h'(e') = u_2;$$

$$t'(e) = u_i \quad \text{if } e \in E(T_i') \text{ and } t(e) = u;$$

$$h'(e) = u_i \quad \text{if } e \in E(T_i') \text{ and } h(e) = u; \text{ for } i = 1,2.$$

For each $v \in V$, $l'(v) = l(v)$ if $l(v) \neq u$;

$$l'(v) = u_1 \quad \text{if } l(v) = u \text{ and } v \in S.$$

$$l'(v) = u_2 \quad \text{if } l(v) = u \text{ and } v \in \bar{S}.$$

It is easy to verify that l' and the V-labelled tree T' satisfy (8.4). Thus, Theorem (8.2) is proved.

9. Proof of (1.11)

Let $[y^0, z^0, w^0]$ be a rational-valued optimum solution to (1.8) where $c = (c_e : e \in E)$ is integer-valued.

(9.0) Starting with $i = 0$, suppose $T, U \in F$, T and U cross and $0 < y_T^i \leq y_U^i$. Then, since F is a crossing family, $T \cap U \in F$ and $T \cup U \in F$. For $S \in F$, define y_S^{i+1} by

$$y^{i+1}_S = \begin{cases} y^i_S + y^i_T & \text{if } S \in \{T \cap U, T \cup U\} \\ y^i_S - y^i_T & \text{if } S \in \{T, U\} \\ y^i_S & \text{otherwise.} \end{cases}$$

It is easy to check that $F(y^{i+1}, e) = F(y^i, e)$ for all $e \in E$. Therefore $[y^{i+1}, z^0, w^0]$ is a feasible solution to (1.8). Furthermore,

$$y^{i+1}f = y^i f + y^i_T[f(T \cap U) + f(T \cup U) - f(T) - f(U)] \le y^i f,$$

by the submodularity of f. Hence $[y^{i+1}, z^0, w^0]$ must also be an optimum solution to (1.8).

Let α be a common denominator of $\{y^0_S : S \in F\}$. Let $u^0 = \alpha y^0$ and for each vector y^{i+1} constructed according to (9.0) let $u^{i+1} = \alpha y^{i+1}$. Since $y^{i+1} \ge 0$, $u^{i+1} \ge 0$ and u^{i+1} is integer-valued. Since $1 \cdot y^{i+1} = 1 \cdot y^i$ we have $1 \cdot u^{i+1} = 1 \cdot u^i$. There can be only a finite number of non-negative integer-valued vectors u having the same sum $1 \cdot u$. Hence there can be only a finite number of distinct vectors in the sequence $\{y^0, y^1, \dots, y^i, y^{i+1}, \dots\}$. Since

$$\sum(y^{i+1}_S|S|^2 : S \in F) = \sum(y^i_S|S|^2 : S \in F) + y^i_T[|T \cap U|^2 + |T \cup U|^2 - |T|^2 - |U|^2]$$
$$> \sum(y^i_S|S|^2 : S \in F),$$

the sequence has only finitely many terms.

(9.1) Therefore there is an optimum solution

$$[y^*_S, z^*_e, w^*_e : S \in F, e \in E]$$

to (1.8) with the property that the family $F^* = \{S \in F : y^*_S > 0\}$ is a cross-free family on V.

(9.2) The vector

$$[y_S, z_e, w_e : S \in F, e \in E],$$

where $y_S = 0$ for $S \in F - F^*$, is a feasible solution to (1.8) whenever

(9.3) $[y_S, z_e, w_e : S \in F^*, e \in E]$

is a feasible solution to the linear program:

(9.4) minimize $\sum(y_S f(S) : S \in F^*) + \sum(a_e z_e - d_e w_e : e \in E)$ by $y \in \mathbf{R}^{F^*}$ and $z, w \in \mathbf{R}^E$ such that $y \ge 0$, $w \ge 0$, $z \ge 0$, and

(9.5) $\forall e \in E,\ z_e - w_e - F^*(y, e) = c_e$;

$$F^*(y, e) = \sum(y_S : S \in F^*, e \in \delta(S)) - \sum(y_S : S \in F^*, e \in \delta(\bar{S})).$$

This is simply the l.p. obtained from (1.8) by suppressing the variables y_S for $S \in F - F^*$.

(9.6) By (9.1) and (9.2), the vector

$$[y_S^*, z_e, w_e^*: S \in F^*, e \in E]$$

is an optimum solution of (9.4), and hence

(9.7) the vector (9.2) is an optimum solution of (1.8) whenever the vector (9.3) is an optimum solution of (9.4).

Denote the system (9.5) by

(9.8) $z - w + yA = c$;

$$A = [a_{Se}: S \in F^*, e \in E]$$

where

$$a_{Se} = \begin{cases} 1 & \text{if } e \in \delta(S), \\ -1 & \text{if } e \in \delta(\bar{S}), \\ 0 & \text{otherwise.} \end{cases}$$

(9.9) Let tree T, and function l from V to $V(T)$, be a V-labelled tree T which represents, as described in Theorem (8.3)–(8.4), the cross-free family F^* on V.

Let H be the graph such that

(9.10) $V(H) = V(T)$, $E(H) = E(T) \cup E$;

(9.11) T is a spanning tree of H;

(9.12) for every $e \in E$, the tail of e in H, $t_H(e)$, is $l(t(e))$ where $t(e)$ denotes the tail of e in G, and the head of e in H, $h_H(e)$, is $l(h(e))$ where $h(e)$ denotes the head of e in G.

By the manner in which T and l represent F^*, for each $S \in F^*$ we have

(9.13) $\delta(S) \cup \{t\} = \delta_H(T(t))$ and

$$\delta(\bar{S}) = \delta_H(\bar{T}(t)),$$

where $\delta_H(\,)$ denotes $\delta(\,)$ with respect to H, where $\delta(S)$ and $\delta(\bar{S})$ are with respect to G, where the other notation is as in (8.5), and where t is the edge of T such that $S = S_t$ as in (8.2).

Hence, the matrix

$$A' = [a'_{te}: t \in E(T), e \in E(H)]$$

where

$$a'_{ie} = \begin{cases} 1 & \text{if } e \in \delta_H(T(i)), \\ 1 & \text{if } e \in \delta_H(\bar{T}(i)), \\ 0 & \text{otherwise}, \end{cases}$$

is the same as the matrix $[J \mid A]$

(9.14) where J is the identity matrix with columns indexed by $E(T)$ and rows indexed by $\{S_i : i \in E(T)\}$ or $E(T)$.

Let M denote the incidence matrix of the graph H. That is

$$M = [m_{ue} : u \in V(H), e \in E(H)]$$

where

$$m_{ue} = \begin{cases} 1 & \text{if } u = t_H(e), \\ -1 & \text{if } u = h_H(e), \\ 0 & \text{otherwise}. \end{cases}$$

(9.15) Clearly we can get row $i \in E(T)$ of A' by adding together the rows of M which are indexed by nodes $u \in T(i)$.

(9.16) Since A' contains the identity matrix J, the rank of A' is $|V(H)| - 1$, and hence the rank of M is at least $|V(H)| - 1$. Since the sum of the rows of M is all zeroes, the rank of M is at most $|V(H)| - 1$.

(9.17) Hence, any row of M is a linear combination of rows of A'. That is,

$$M = DA' = D[J \mid A] = [D \mid DA]$$

where D consists of the columns of M which are indexed by $E(T)$.

By (9.8) and (9.14), we may express the linear program (9.4) in the form:

(9.18) minimize $yf - zc - wd$
by $y \in \mathbf{R}^{E(T)}$, $z \in \mathbf{R}^E$, $w \in \mathbf{R}^E$
satisfying $z \geq 0$, $w \geq 0$, $yJ \geq 0$, and $z - w + yA = c$.

The dual l.p. of (9.18) is

(9.19) max cx
where $x \in \mathbf{R}^E$, $u \in \mathbf{R}^{E(T)}$
satisfy $d \geq x \geq a$, $u \geq 0$, and

(9.20) $Ju + Ax = [J \mid A]\binom{u}{x} = f$.

Multiplying equation (9.20) by D we get

(9.21) $M\binom{u}{x} = Df$.

The linear program (9.19) with (9.20) replaced by the equivalent (9.21) is a familiar optimum network flow problem, which we denote by (R). As we mentioned in Section 2, it is well-known that, for integer-valued c, the dual l.p. of (R) has an integer-valued optimum solution if it has an optimum solution. The dual l.p. of (R) is

(9.22) minimize $\pi Df + za - wd$
by $\pi \in \mathbf{R}^{E(V)}, z \in \mathbf{R}^E, w \in \mathbf{R}^E$
satisfying $z \geq 0, w \geq 0, \pi DH \geq 0, z - w + \pi DA = c.$

By (9.6), (9.8) has an optimum solution. Hence (9.19) and (R) have optimum solutions. Hence (9.22) has an integer-valued optimum solution, say $[\pi', z', w']$.

Clearly, since D is integer-valued,

(9.23) where $y' = \pi' D$, $[y', z', w']$ is an integer-valued optimum solution of (9.18), i.e., of (9.4).

(9.24) Therefore, by (9.2), we have an integer-valued optimum solution to (1.6). $\square$

10. Everything above

(10.0) For any polyhedron $P \subseteq \mathbf{R}^E$, the *dominent* of P is defined as $P + \mathbf{R}^E_+ = \{w : w \geq x$ for some $x \in P\}$. One purpose now is to describe the dominent of the polyhedron P of any system of the form (1.6).

(10.1) For example we have seen in (6.3) that one such P is the bounded polyhedron, say $P(G, k)$, whose set of vertices is the set of incidence vectors of directed-cut k-coverings of G. For a given graph $G = (V, E)$ and a given integer $k \geq 0$.

(10.2) it follows immediately from (6.3) that where $F_1 = \{\delta(R): R \in D(G)\}$ is the family of directed cuts of graph $G = (V, E)$, $f(S) = k$ for $S \in F_1$, $F_2 = \{\{e\}: e \in E\} - F_1$, $f(S) = 0$ for $S \in F_2$, $F = F_1 \cup F_2$, and $a = (a_e : e \in E)$ is all ones, then $P(G, k)$ is the P of (10.17) below.

We will see that

(10.3) $P(G, k) + \mathbf{R}^E_+$ is defined by the system in (10.18) below.

(10.4) Another purpose now is to place the polyhedra P of (1.6) in a broader setting. We say that a polyhedron $P \subset \mathbf{R}^E$ is *box* TDI if P intersected with any "box", $\{x \in \mathbf{R}^E : d \leq x \leq a\}$, d and $a \in (\mathbf{R} \cup \{\pm\infty\})^E$, is totally dual integral. We say that a linear inequality system is *box* TDI if it together with any upper and lower bounds on the individual variables is totally dual integral. Our main theorem (1.11) states that systems (1.6), and hence their polyhedra, including the above $P(G, k)$, are box TDI.

(10.5) For any polyhedron $P = \{x \in \mathbf{R}^E : \sum (a_j x_j : j \in E) \geq b\}$, a_j and b being appropriate vectors, and for any function, $\phi: E' \to E$, from a finite set E' into E, we say that

$$P' = \left\{ x \in \mathbf{R}^{E \cup E'} : \sum (a_j x_j : j \in E) + \sum (a_{\phi(j')} x_{j'} : j' \in E') \geq b \right\}$$

is *obtained from P by duplicating*, $|\{j' \in E' : \phi(j') = j\}|$ times, the variable x_j, for all $j \in E$.

(10.6) Clearly, every polyhedron P', obtained from a P of (1.6) by duplicating variables, is itself given by a system of the form (1.6), where the graph G' which gives P' is obtained from the graph G which gives P simply by "duplicating" edges so that each edge $j \in E'$ has the same head and tail as edge $\phi(j) \in E$. Hence, every polyhedron P', obtained from the polyhedron P of a system (1.6) by duplicating variables, is box TDI.

In another paper on the box TDI property we prove the following:

(10.7) Theorem. *Any box* TDI *polyhedron is defined by a system $Ax \geq b$ where A is a matrix such that every entry is* 0, 1, *or* -1.

(10.8) Theorem. *Any polyhedron obtained from a box* TDI *polyhedron by duplicating variables is itself box* TDI.

Using (10.8), we now prove:

(10.9) Theorem. *Where $P = \{x : Ax \geq b\} \neq \emptyset$ is box* TDI, *the dominant of P is the set of points w such that $\pi A w \geq \pi b$ for every integer-valued $\pi \geq 0$ such that πA is* 0, 1 *valued.*

Proof. A vector w is such that $w \geq x$ for some $x \in P$ iff

(10.10) $\min\{1 \cdot x' : x' \geq 0, \; -x \geq -w, \; Ax - Ax' \geq b\} \leq 0.$

By the l.p. duality theorem, (10.10) holds iff

(10.11) $\max\{-tw + \pi b : t \geq 0, \; \pi \geq 0, \; -t + \pi A = 0, \; \pi A \leq 1\} \leq 0.$

Since, by (10.8), the constraint system of (10.10) is TDI, and since the objective function has integer coefficients, the maximum in (10.11) is achieved by an integer-valued (t, π). Hence, (10.11) holds iff

(10.12) $tw \geq \pi b$ for every integer-valued (t, π) such that $\pi \geq 0$ and $0 \leq t = \pi A \leq 1$. $\square$

Of course, by (10.6), we have Theorem (10.9) for the case where $Ax \ge b$ is of the form (1.6) without using (10.8).

(10.13) It is trivial to show that any face of a box TDI polyhedron is box TDI.

(10.14) By (4.11), the convex hull, say P, of the vectors of largest common independent sets of two matroids M_1 and M_2 on set E, is a face of $P_1 \cap P_2$ as defined by (4.1) and (4.10). Hence, P is box TDI, and so, by (10.9),

(10.15) Theorem. *$P + \mathbf{R}^E_+$ is the set of solutions x of the system*

$$\forall S \subseteq E,\ x(S) \ge h(S),$$

where

$$\begin{aligned} h(S) &= \min\{x(S) : x \in P\} \\ &= \min\{|S \cap J| : J \text{ is a largest common independent set} \\ &\qquad \text{of } M_1 \text{ and } M_2\}. \end{aligned}$$

Theorem (10.15), which has also been proved by W.H. Cunningham [3], answers affirmatively a conjecture of Ray Fulkerson [11].

The following result was suggested to us by our co-worker in polyhedral combinatorics, Gilberto Calvillo.

(10.16) Theorem. *Let F be any family of subsets of E. Let $f(S)$, $S \in F$, be any real valued function of F. Let $a = (a_e : e \in E) \in \mathbf{R}^E$.*

(10.17) *Let $P = \{x \in \mathbf{R}^E : x \le a;\ \forall S \in F, x(S) \ge f(S)\} \ne \emptyset$.*

(10.18) *Then $P + \mathbf{R}^E_+ = \{x \in \mathbf{R}^E : T \subseteq S \in F, x(S - T) \ge f(S) - a(T)\}$, where $a(T) = \sum (a_e : e \in T)$.*

Proof. By a version of Farkas lemma, for any polyhedron $P = \{x \in \mathbf{R}^E : Ax \ge b\} \ne \emptyset$ where A is rational, we have a finite set Π of rational vectors $\pi \ge 0$, $\pi A \ge 0$, such that

$$P + \mathbf{R}^E_+ = \{x \in \mathbf{R}^E : \forall \pi \in \Pi, (\pi A)x \ge \pi b\}.$$

Clearly, we may consider each $\pi \in \Pi$ to be integer-valued.

Where $Ax \ge b$ is the system defining P in (10.17), each $\pi \in \Pi$ has a component, say π_S, for the inequality

$$x(S) \ge f(S), \quad S \in F; \tag{S}$$

and has a component, say π_e, for the inequality

$$-x_e \ge -a_e, \quad e \in E. \tag{e}$$

We prove the theorem by showing that each $(\pi A)x \geq \pi b$, is a non-negative combination of inequalities of the form

$$x(S-T) \geq f(S) - a(T), \; T \subseteq S \in F. \qquad (S, T)$$

Clearly, those inequalities where $T = S$ are not needed since $P \neq \emptyset$.

Think of $(\pi A)x \geq \pi b$ as obtained by adding together a family $\Pi_F \cup \Pi_E$ of inequalities where Π_F consists of π_S copies of inequality (S), for each $S \in F$, and Π_E consists of π_e copies of inequality (e), for each $e \in E$. Since the e component of πA is

$$-\pi_e + \sum(\pi_S : e \in S) \geq 0,$$

clearly there exists a mapping ϕ of Π_E to Π_F such that, for each $i \in \Pi_F$, $\phi^{-1}(i)$ contains at most one copy of each inequality (e), and, where i is a copy of inequality (S), we have $T \subseteq S$ where T is the set of elements e such that $\phi^{-1}(i)$ contains a copy of inequality (e).

For each $i \in \Pi_F$, by adding together inequality i and inequalities $\phi^{-1}(i)$, we get an inequality, say i', of the form (S, T). Adding together all of the inequalities i', for $i \in \Pi_F$, we get $\pi A x \geq \pi b$. $\square$

References

[1] E. Balas, A linear characterization of permutation vectors, Management Science Research Report No. 364, Carnegie-Mellon University, Pittsburgh, Penn. (1975).

[2] W.H. Cunningham, *A combinatorial decomposition theory*, Doctoral Thesis, University of Waterloo, Waterloo, Ontario (1973).

[3] W.H. Cunningham, An unbounded matroid intersection polyhedron, Technical Report No. 234, December 1975, Johns Hopkins University, Department of Applied Mathematics (to appear).

[4] J. Edmonds, Minimum partition of a matroid into independent subsets, *J. of Research, Section B, N.B.S.* 69B (1965) 67–72.

[5] J. Edmonds, Matroid partition, in *Math. of the Decision Sciences, Part I* (A.M.S., Providence, RI, 1968) 335–345.

[6] J. Edmonds, Submodular functions, matroids and certain polyhedra, in *Combinatorial Structures and their Applications* (Gordon and Breach, 1970) 69–87.

[7] J. Edmonds, Matroids and the greedy algorithm, *Math. Programming* 1 (1971) 127–136.

[8] J. Edmonds, Matroid intersection, unpublished paper.

[9] J. Edmonds and R. Giles, Polymatroid theory (to appear).

[10] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, NJ, 1962).

[11] D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra, *Math. Programming* 1 (1971) 168–194.

[12] R. Giles, *Submodular functions, graphs and integer polyhedra*, Doctoral Thesis, University of Waterloo, Waterloo, Ontario (1975).

[13] L. Lovász, On two minimax theorems in graph theory (to appear).

[14] C. Lucchesi and D. Younger (to appear).

[15] W.R. Pulleyblank, *Faces of matching polyhedra*, Doctoral Thesis, University of Waterloo, Waterloo, Ontario (1973). Also: Edmonds and Pulleyblank, Optimum matchings and polyhedral combinatorics, Johns Hopkins University Distinguished Lectures in Applied Mathematics, May 1975, to be published by the Johns Hopkins University Press.

[16] K. Vidyasankar and D. Younger, A minimax equality related to the longest directed path in an acyclic graph, *Canad. J. of Math.* 27 (1975) 348–351.

Annals of Discrete Mathematics 1 (1977) 205–220

HOW CAN SPECIALIZED DISCRETE AND CONVEX OPTIMIZATION METHODS BE MARRIED?*

A.M. GEOFFRION

Western Management Science Institute, University of California, Los Angeles, CA, U.S.A.

Numerous practical problems involve both logical design choices and continuous-valued decision variables which are predicated in some manner on the logical design. For instance, industrial scheduling problems usually involve both sequencing and the determination of how continuously divisible resources should be applied for the chosen sequence, and network synthesis problems involve both the logical design of the network and the programming of flows for the chosen design. Many such problems which are difficult to solve directly as a whole have the tantalizing properties that (a) specialized algorithms (discrete or combinatorial) are available for close relatives of the logical design aspect of the problem, and (b) for any particular logical design the resulting continuous optimization problem can be solved by an available convex programming method (usually by LP or a network flow technique). This raises the question of how the two specialized types of algorithms can be married to provide an effective overall approach to the problem. Several possible kinds of marriages are surveyed and attractive opportunities for further research are pointed out.

1. Introduction

Some of the most difficult yet important potential applications of optimization are to decision and design problems which involve a mixture of both discrete and continuous-valued choices. It is unfortunate that the mathematical apparatus and algorithmic approaches applicable to the discrete aspect of such problems are usually entirely different from and incompatible with those applicable to the continuous aspect. The dissimilarities between discrete/combinatorial optimization and linear/nonlinear programming are many and profound. Consequently, the state-of-the-art for such hybrid problems is well behind that for problems which involve only discrete choices or only continuous-valued choices. With but few exceptions, the current practice is to adopt a discrete or combinatorial approach with an approximation which essentially submerges the continuous choice aspect of the problem, or to do the converse, or to adopt a heuristic approach which treats both aspects of the problem more evenhandedly.

The purpose of this paper is to begin the systematic study of methods by which effective hybrid algorithms can be developed for hybrid problems. The prospects for success seem brightest for a broad class of problems dubbed "discrete/convex

* This paper was partially supported by the National Science Foundation and by the Office of Naval Research, and was presented at the Workshop on Integer Programming near Bonn, Germany, September 8–12, 1975.

programs." We define this class, survey its applications, describe four promising approaches to the development of applicable hybrid algorithms, and finally conclude with an indication of attractive opportunities for further research.

1.1. *Definition of discrete/convex programming*

By a *discrete/convex program* we mean an optimization problem of the form

$$\begin{aligned} &\operatorname*{Min}_{\delta, x}\; c_\delta + f_\delta(x) \\ &\text{s.t.}\quad \delta \in \Delta,\ x \in X_\delta, \end{aligned} \tag{DC}$$

where Δ is a *finite* set of possible discrete choices or logical designs δ, and X_δ is a *convex* set of possible continuous choices or activities x associated with any given δ. The objective function distinguishes the direct cost of δ, c_δ, from the cost $f_\delta(x)$ of the activities carried out under δ. The asymmetry of the notation in δ and x reflects the fact that, in many of the applications we have in mind, the choice of x is predicated on the choice of δ but not conversely; that is, the very domain of x may depend on δ whereas the domain of δ can always be described independently of x. More specifically, we presume that (DC) satisfies these two properties:

Property 1. For any fixed δ in Δ, $f_\delta(\cdot)$ is convex on X_δ and its minimum can be computed with reasonable efficiency by a known convex programming algorithm (e.g., by LP, NLP, a network flow method, etc.)

Property 2. A reasonable efficient discrete or combinatorial optimization algorithm is known for some problem related to (and hopefully a reasonable approximation of)

$$\operatorname*{Min}_{\delta \in \Delta}\; c_\delta + v(\delta), \quad \text{where } v(\delta) \triangleq \operatorname*{Inf}_{x \in X_\delta} f_\delta(x) \tag{D}$$

Problem (D) obviously is equivalent to (DC): it is infeasible or has unbounded optimal value if and only if (DC) does; and if δ^0 is optimal (ε_1-optimal) in (D) and x^0 is optimal (ε_2-optimal) in the "inner" problem defining $v(\delta^0)$, then (δ^0, x^0) is optimal ($\varepsilon_1 + \varepsilon_2$-optimal) in (DC).[1] Notice that Property 1 assures the relatively easy evaluation of $v(\delta)$. Exactly what relative of (D) for which a discrete or combinatorial algorithm is available is deliberately left unspecified in Property 2. Usually $v(\cdot)$ must be approximated by a much simpler function in such an algorithm, and sometimes c_δ or even Δ must also be approximated. The intent of Property 2 is simply to focus on applications where the discrete aspect of the problem is tractable provided suitable approximations are made to submerge the continuous aspect.

One further comment must be made about (DC): although X_δ will necessarily be a subset of a finite-dimensional vector space, no such restriction need be imposed on Δ. In some applications δ will be a map of one finite set into another, or some

[1] See, e.g., [15, Theorem 2] (where (D) would be called the "projection" of (DC) onto δ).

other combinatorial object, rather than a tuple of real numbers. It is not the structure of the space in which Δ dwells, but rather the logical structure of Δ itself (in addition to finiteness) which permits mathematical manipulations involving δ to be carried out. Of course (DC) could always be reformulated so that δ is replaced by integer-valued indicator variables. However, in most applications such an artifice serves only to obscure the natural structure of Δ and to cause an excessive increase in representational complexity or size or both.[3] It therefore seems wise *not* to insist that (DC) be stated as a conventional mathematical programming problem in real variables and equality or inequality constraints.

2. Some applications

Here we survey briefly some of the principal types of applications which fall within the domain of discrete/convex programming as defined above.

2.1. *Production scheduling* [21, 24, 28, 29]

Setup and sequence-dependent changeover costs, minimum batch sizes, precedence constraints, and crew integrity are some of the factors which remove many production scheduling problems from the realm of ordinary linear or nonlinear programming. The logical design δ typically determines which jobs are to be done in what order on which machines (or machine configurations), and possibly which crew will handle each setup. The activity vector x then determines, for a given δ, the timing and quantities of each run, the allocation of divisible resources to job activities, and so on.

An algorithm in keeping with Property 1 is likely to be of LP type, possibly with some nonlinear costs, while combinatorial algorithms in keeping with Property 2 abound (but with only limited success) in the literature on machine/job shop scheduling/sequencing [6, 7]. Example 1 describes a case where a successful partnership was achieved between linear programming and a quadratic assignment algorithm (see Section 3.1).

2.2. *Network design* [1, 2, 3, 4, 5, 12, 13, 22]

Many problems connected with the design or modification of communication networks and transportation networks can be posed as discrete/convex programs. The discrete design δ may select nodes for the installation of facilities — multiplexers, concentrators, or interface message processors in computer communication networks, junctions in pipeline networks, interchanges in highway networks,

[3] See Example 1 below, and think of the futility of attempting to express many realistic scheduling and sequencing problems as integer linear programs.

and so on. A design δ may also select connecting links from a finite list of possibilities, both in terms of which nodes are to be connected and in terms of the capacity of the connection (there are standard transmission speeds for communication lines, standard sizes for gas and oil pipelines, only a few choices for the number of lanes of a highway, etc.). The choice of discrete design requires that due consideration be given to its impact on the flows in the network. Differences in unit flow costs, delays due to congestion, and demand elasticity all tend to render flow prediction a nontrivial problem even when δ is fixed (see [13] for a discussion of the influence of cost and congestion on the utilization of store-and-forward communication networks, and [9, 33] for a discussion of equilibrium flows in transportation networks). The activity vector x represents, of course, the flows in a network.

Network flow algorithms are obviously the most natural choice for the task posed by Property 1, particularly since their power has increased dramatically during the last few years. Convex cost functions occur when congestion delays are taken as the criterion [13]. A variety of discrete optimization algorithms have potential for Property 2: minimum spanning trees [4] when the network must have a tree structure, set covering [34] for emergency service networks, generalized assignment [31] when peripheral facilities must be linked directly to fixed service facilities, and so on. Example 2 describes an application where a multicommodity flow algorithm can be combined with a knapsack algorithm (see Section 3.2).

2.3. *Physical distribution system planning* [19, 20, 25]

In distribution system planning problems the discrete design δ determines the geographic location of plants and/or warehouses, and possibly also the all-or-nothing assignment of customers to these facilities for each integral bundle of products. The activity vector x corresponds to product flows. This class of models is conceptually close to network design as discussed above, but has enough distinguishing characteristics (such as the absence of link capacities and the presence of facility capacities and economies-of-scale) that separate treatment is warranted.

2.4. *Facilities layout* [10, 23]

Facilities layout problems occur on a hierarchy of scales. On a global scale, in which cities should the various facilities of a firm be located? Within a given city, which sub-facility should be located in each available building? Within a given building, which department or operating unit should be located on each floor and in each work area? Within a given work area, what should be the layout of the various pieces of equipment? The problem appears to be a combinatorial one, but flows and communications can be influenced by locational layout and often need to be considered jointly. Locational layout would be specified by δ and x would specify flows and communications.

Example 3 describes an application where linear programming for flow/communications is combined with a quadratic assignment algorithm for the layout choices (see Section 3.3).

2.5. *Other applications*

There are many other applications which can be modeled as discrete/convex programs. One interesting class is that of selecting and sequencing interdependent capital investment projects (for hydroelectricity, manufacturing capacity expansion, etc.). The logical design δ would determine which projects are selected and their sequence of execution, while x would determine the details of project timing and how the system corresponding to a given δ is operated over time. A particularly nice case is developed in [8], where a dynamic programming approach was derived for (D) itself that can be used for a variety of different "operating cost submodels" specified by X_δ and $f_\delta(x)$.

Another important class of applications for discrete/convex programming is transport scheduling. The problem here is different from the transportation network design problems discussed earlier because the major emphasis is on how fleet vehicles (planes, ships, trains, pool trucks, etc.) should move over an established transportation network in response to demands for transport. The possible sequences of moves for each vehicle comprises the combinatorial aspect of the problem, while the exact timing of the moves and the determination of passenger/cargo patronage comprises the continuous aspect. It is usually essential to consider both aspects together since patronage adjusts to the frequency and timing of transport service. See, for instance, [30] for a treatment of the problem in the context of airline routing; the evaluation of $z(\delta)$ is a linear programming problem which determines the maximum profit loading of available passengers on flights.

3. Computational approaches

We now describe four promising generic computational approaches to the development of hybrid algorithms for discrete/convex programming. They are: (i) combinatorial seeding with local convex enumeration, (ii) generalized branch-and-bound, (iii) cyclic marginal optimization over δ and x, and (iv) improving approximations to (D).

3.1. *Combinatorial seeding with local convex enumeration*

By Property 2, a discrete optimization algorithm is available for some relative of (D). Let δ^0 be the resulting approximation to an optimal choice for δ. Now use δ^0 as a "seed" to be improved, if possible, via "low order" changes evaluated by the convex programming algorithm postulated by Property 1. What constitutes a low order change depends on the structure of Δ; for instance, if δ were a binary n-tuple the order of change might be measured as the number of components whose values are altered. It is helpful but not necessary for Δ to be a subset of a metrizable space. Sometimes it is convenient to use the term "neighbor" for any modification of δ that qualifies as being of acceptably low order. The emphasis on low order changes is designed, of course, to restrict the magnitude of the local enumeration task. Generally one wants the allowable order of change to be sufficiently low that local enumeration is computationally practical, yet sufficiently high that an improved logical design will be found if one exists.

This approach is pictured informally in Fig. 1. It is understood that the seed is not actually replaced as the incumbent until one of its neighbors proves to yield a superior feasible solution of (DC). Termination occurs when no neighbor of the current incumbent is superior; the higher the allowable order of change the stronger the degree of local optimality at termination.

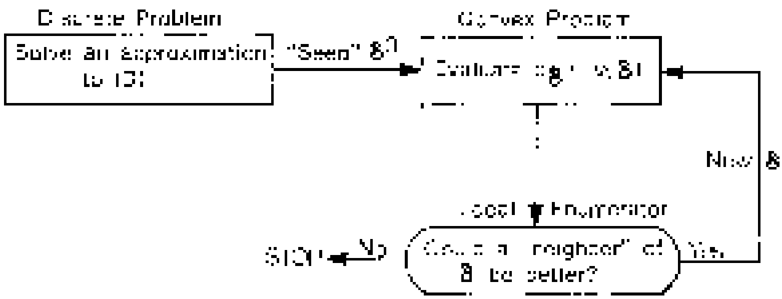


Fig. 1

A variant would be to generate several seeds from (D) rather than just one, as by solving several approximations to (D) or by finding several suboptimal solutions to a single approximation.

This approach has familiar analogs in the literature on heuristic programming. See [14, Chapter 9] and [27]. See also [32] for a highly successful application to gas pipeline network design that has since been adapted and used extensively for computer communication network design (e.g., [17]).

The author has had very satisfactory experience with this approach in the context of scheduling parallel chemical reactors with product-dependent changeover costs. This application is now briefly reviewed.

Example 1. A changeover scheduling problem [21]. Several independent continuous process facilities or flow shop production lines are arranged in parallel. Each can make (process) some subset of products with production rates that may vary from line to line, but that are reasonably proportional from line to line (as would be the case when lines are similar except for their scale of implementation or their basic cycle time). Each line has a linear production cost for each product it can make, and a possibly different changeover cost between each pair of products. The changeover cost matrices are reasonably proportional across lines. A number of independent production orders are given, each of which specifies a minimum and maximum production quantity, an earliest start date, and a due date. Violation of either date incurs a per diem cost penalty. Splitting production orders is allowed. It is desired to find a production schedule — which line produces how much of what and when — that fills the production orders at minimum total cost over a scheduling horizon of fixed (but somewhat flexible) length.

In this application, δ gives the sequence of production runs specified as to product and line but *not* fully specified as to duration. Durations are given by x. Property 1 holds because, when δ is fixed, the optimal choice of x may be determined by solving a linear program. The LP balances production costs (exclusive of changeover charges) against penalties associated with any violations of earliest start and due dates. Property 2 holds because (D) can be approximated quite well by a quadratic assignment problem of reasonable size.

An LP code and quadratic assignment code were combined in the manner of Figure 1. The definition used for "neighbor" was that any single production run may be moved to another position on the same or another line, and any two production runs may be interchanged.

A real application was made to the monthly scheduling of a complex of six chemical reactors. A three month independent parallel test showed that the program was able to achieve considerably better solutions than (experienced) manual schedulers. The program has since been installed on the firm's computer and is being used routinely [71].

3.2. *Generalized branch-and-bound*

The essential concepts of branch-and-bound, currently the dominant approach to integer programming, require very little mathematical structure and are quite broad enough to encompass discrete/convex programming. The framework of [22] will serve nicely with only the obvious notational changes to phrase it in terms of (DC) rather than in terms of mixed integer linear programming. It is also advisable to generalize the notion of "relaxation," whence nearly all bounds are obtained in branch-and-bound methods, to the following: a minimizing problem (P_R) is said to be a *relaxation* of a minimizing problem (P) if the feasible region of (P_R) contains that of (P) and if the objective function of (P_R) is less than or equal to that of (P) everywhere on the latter's feasible region. This generalized definition requires an obvious modification to property R3 and fathoming criterion FC3 in [22] in order to reflect the fact that an optimal solution of (P_R) is not optimal in (P) unless it is feasible in (P) *and* yields the same objective function value for both problems (although an ε-optimality statement can still be made if the very last condition fails). [22] will be sufficiently accessible to most readers that the algorithmic framework of Section II therein, as generalized to (DC), need not be given in detail here.

So far, no use whatever has been made of Properties 1 and 2. The principal way of doing so is to select a type of relaxation which permits advantage to be taken of one or the other or both of these properties when trying to fathom the candidate problems (alias nodes or sub-problems). There are two major types of relaxations used in mixed integer linear programming, both of which can be generalized to apply to candidate problems derived from (DC) provided certain conditions hold: relaxations based on direct convexification of the decision domain of the candidate problems (as by allowing integer variables to take on continuous values), and Lagrangean relaxation of selected constraints [18]. Suppose that candidate problems are derived from (DC) by partially specifying certain components of δ (we presume, as seems permissible for most potential applications, that the structure of Δ renders this prescription meaningful). An obvious difficulty with such candidate problems is that the very notion of convexification in the domain of δ is not meaningful unless δ inhabits a vector space, which definitely is not the case in many applications of interest (e.g., Example 1). Moreover, the mathematical operation of Lagrangean relaxation requires X_δ to be expressible at least partially in terms of conventional real-valued equality or inequality constraints. The first difficulty can be skirted if necessary by convexifying not in the *domain* of δ, but rather in the *range* spaces associated with δ — the range of the real-valued function c_δ, and of the point-to-set map X_δ. The second difficulty apparently cannot be skirted.

There is a striking relationship between the two types of relaxation just discussed. It was shown in [18] that, for mixed integer linear programs, the best possible Lagrangean relaxation is equivalent in a natural sense to a corresponding convexification in the domain of the decision variables and also to a corresponding convexification in the range space of the objective function and Lagrangeanized constraints. The analysis can be generalized. Dropping the assumption that all functions are linear invalidates the equivalence to convexification in domain space but does not invalidate the equivalence to convexification in range space. The latter equivalence even remains true when δ is no longer taken to dwell in a finite vector space, and when the constraining conditions other than those being Lagrangeanized are no longer expressible as conventional real-valued constraints. This is a consequence of the fact that many basic results of Lagrangean duality theory require virtually no assumptions at all on the domains of the functions (e.g., [16, Lemmas 3, 4 and 5]). A formal proof of the basic equivalence between the best Lagrangean relaxation and problem convexification in range space can be found in [26, Lemma 2.2].

In particular applications one seeks to apply the convexification or Lagrangean relaxation devices just discussed or possibly some other device, in order to obtain candidate problem relaxations which Properties 1 and/or 2 render tractable. The following example illustrates a situation in which this can be done.

Example 2. Network expansion with a budget constraint. This problem is a capacitated version of the one treated in [2]. A conventional multicommodity network is given with capacitated links, a known flow requirements matrix, and linear flow costs. A number of possible new links have been proposed, each with a given flow capacity, linear flow cost, and fixed capital cost. What is the optimal subset of new links which reduces the total cost of the optimal flow as much as possible without exceeding a given maximum authorized capital expenditure?

The problem can be stated mathematically as follows in an obvious notation where ij refers to the particular commodity which flows from the ith to the jth node, A is the set of existing links, B is the set of possible new links, and D is the capital budget.

$$\operatorname*{Min}_{x,\delta} \sum_{ij} \sum_{kl} c^{ij}_{kl} x^{ij}_{kl} \quad \text{subject to} \tag{P}$$

$$\sum_{k} x^{ij}_{lk} - \sum_{k} x^{ij}_{kl} = \begin{Bmatrix} -r_{ij} & \text{if } l = i \\ +r_{ij} & \text{if } l = j \\ 0 & \text{otherwise} \end{Bmatrix} \text{for all } ij, \tag{1}$$

$$\sum_{ij} x^{ij}_{kl} \leq b_{kl} \qquad \text{for all } kl \in A, \tag{2}$$

$$\sum_{ij} x^{ij}_{kl} \leq b_{kl}\delta_{kl} \qquad \text{for all } kl \in B, \tag{3}$$

$$\sum_{kl \in B} d_{kl}\delta_{kl} \leq D, \tag{4}$$

$$x^{ij}_{kl} \geq 0, \qquad \text{for all } ij \text{ and } kl \in A \cup B, \tag{5}$$

$$\delta_{kl} = 0 \text{ or } 1, \qquad \text{for all } kl \in B. \tag{6}$$

This is a mixed integer linear programming problem which, for reasonable numbers of potential new links (not much more than a hundred, say), should be tractable by branch-and-bound if the main candidate problem relaxation is chosen suitably. The usual LP relaxation, obtained by allowing the free binary variables to be fractional, is not a multicommodity flow problem; efficient specialized multicommodity flow algorithms cannot be used and one must fall back to general linear programming algorithms. An attractive alternative to the usual LP relaxation is to employ a "tandem" Lagrangean relaxation. This will be illustrated on the full problem (P) as stated above since the candidate problems are of the same mathematical form so long as conventional dichotomous branching is used.

Let $\mu^0 \geq 0$ be the analyst's best guess concerning the marginal value to (P) of increasing the budget D by one dollar. Solve the relaxation of (P) which results when (4) is Lagrangeanized using μ^0 and (6) is convexified in the usual way. This is equivalent to an ordinary classical multicommodity flow problem because the δ_{kl} variables can be eliminated analytically (solve for δ_{kl} from (3), which must hold with equality in an optimal solution):

$$\operatorname*{Min}_{x} \sum_{i} \sum_{kl} c^i_{kl} x^i_{kl} + \mu^0 \Big(\sum_{kl \in B} d_{kl} \sum_{i} \frac{1}{b_{kl}} x^i_{kl} - D \Big) \qquad (\mathrm{MF}_{\mu^0})$$

s.t. (1), (2), (5)

$$\sum_{i} x^i_{kl} \leq b_{kl} \quad \text{for all } kl \in B. \qquad (3)'$$

Let x^0 be an optimal solution, and let ξ^0_{kl} be the optimal multipliers corresponding to (3)'. It can be shown that

$$\lambda^0_{kl} \triangleq \xi^0_{kl} + \mu^0 \Big(\frac{d_{kl}}{b_{kl}} \Big) \quad \text{for all } kl \in B \qquad (7)$$

is a set of optimal multipliers corresponding to (3) in the relaxed version of (P) prior to analytic reduction to (MF_{μ^0}). Now solve a second relaxed version of (P) in which (3)' is appended and λ^0 from (7) is used to Lagrangeanize (3):

$$\operatorname*{Min}_{x,\delta} \sum_{i} \sum_{kl} c^i_{kl} x^i_{kl} + \sum_{kl \in B} \lambda^0_{kl} \Big(\sum_{i} x^i_{kl} - b_{kl}\delta_{kl} \Big) \qquad (\mathrm{PR}_{\mu^0})$$

s.t. (1), (2), (3)', (4), (5) and (6).

Evidently this problem can be solved independently for x and for δ. It is easy to show that x^0 from (MF_{μ^0}) is also optimal here, leaving just the binary knapsack problem

$$\operatorname*{Max}_{\delta} \sum_{kl \in B} (\lambda^0_{kl} b_{kl}) \delta_{kl} \quad \text{subject to (4) and (6)} \qquad (\mathrm{K}_{\mu^0})$$

as the only work necessary to solve the second relaxation (PR_{μ^0}). Methods are available which can solve (K_{μ^0}) very efficiently even with several hundred binary variables.

In summary, a tandem relaxation of (P) has been proposed which requires the solution of one ordinary multicommodity flow problem (cf. Property 1) and one binary knapsack problem over the possible new links (cf. Property 2). Both Properties 1 and 2 are exploited. An otherwise conventional branch-and-bound procedure can be built around this tandem relaxation. How well such a procedure would function depends on how good the resulting bounds are. This has not been tested experimentally, but it can be observed from the known theory of Lagrangean relaxation [18] that the lower bound produced by this tandem relaxation has the potential of being superior to that provided by the usual LP relaxation (in which (6) is convexified). It all depends on the choice of μ^0. If μ^0 happens to have the same value as an optimal multiplier of (4) in the usual LP relaxation, then the bound produced by (MF_{μ^0}) will coincide with that of the usual LP relaxation and the second bound obtained with the help of (K_{μ^0}) will usually be still better (it cannot be worse).

It may be worthwhile to iterate on the choice of μ^0. There are at least two conspicuous ways to do this. One is to perform a one-dimensional (unimodal) search for the value of μ which leads to the *highest* optimal value of (MIC). This is particularly easy to do if a parametric multicommodity flow algorithm is available which accommodates a single linear parameter in the objective function (the cost coefficients of the links in set B are $c_{ij}^k - \mu d_{ij}/b_0$). This search is equivalent to solving the partial dual of the usual LP relaxation in which only the budget constraint (4) is dualized. The second way to find an improved μ is to *feed back* the budget constraint multiplier from (K,r) with (6) convexified.

3.3. *Cyclic marginal optimization over δ and x*

In some applications, Property 2 permits (DC) to be optimized with any fixed x. Then it is natural to think of seeking an optimum of (DC) by first optimizing over x with some fixed δ, then optimizing over δ with the resulting x, then by optimizing over x again with the new resulting δ, and so on. A monotonely improving succession of feasible solutions will be found by such a *cyclic marginal optimization* approach until a "marginally optimal" solution is found after which the marginal solutions in x and δ begin to repeat. Marginal optimality is an obvious necessary condition for global optimality, but whether it is sufficient depends upon the structure of the problem.

This general approach is, of course, far from novel (e.g., [35, p. 111]).

The following example illustrates a plausible application of this approach in which the discrete and convex marginal optimization problems are, respectively, a quadratic assignment problem and a linear program.

Example 3. A facility assignment problem. A firm has a number of indivisible facilities and a number of distinct locations to which they could be assigned. The firm carries on a number of different activities, each of which imposes its own requirements for "traffic" between the facilities. These requirements are sufficiently dissimilar, and the traffic costs are sufficiently high, that the assignment of facilities to locations materially influences the most profitable mix of activities. It is therefore appropriate to optimize jointly the facility location assignments and activity mix.

We adopt the following notations and assumptions:

x_k the level of the kth activity of the firm,
$Ax \leq b$ the constraints specifying the set of possible activities
$x \geq 0$ (independent of the facility location assignments),
p_k the net profit per unit of activity k exclusive of traffic costs,
q_{ij}^k the amount of traffic between facilities i and j incurred for each unit of activity k,
c_{lm} the cost per unit of traffic between locations l and m.

a_{il} the cost associated with assigning facility i to location l (can be ∞ to indicate an impossible assignment).

δ a mapping of facilities into locations; $\delta(i) = l$ means that δ assigns facility i to location l.

Then the problem can be written:

$$\operatorname*{Min}_{x,\delta} \sum_i \sum_j \Big(\sum_k q_{ijk} x_k\Big) c_{\delta(i)\delta(j)} + \sum_i a_{i\delta(i)} - \sum_k p_k x_k$$

s.t. $Ax = b$,

$x \geq 0$,

δ a 1-1 mapping.

It is evident that this is an ordinary quadratic assignment problem for fixed x and an ordinary linear program for fixed δ, and hence a plausible candidate for cyclic marginal optimization. This approach has not been tested computationally.

3.4. *Improving approximations to* (D)

The essential idea of this computational approach is to generate a sequence of approximations to (D) which are improving in the sense that their solutions tend to converge to an optimal solution of (D) itself. Property 1 comes into play in the course of evaluating the performance $v(\delta^K)$ of the solution δ^K of the Kth approximation $(\hat{D})^K$. Of course, the form of $(\hat{D})^K$ must be compatible with the scope of Property 2. A rule must be specified to prescribe how $(\hat{D})^K$ is to be generated based on knowledge of δ^k and x^k obtained from previous ($k = 1, \ldots, K-1$) evaluations of $v(\delta^k)$ and $(\hat{D})^k$. See Fig. 2.

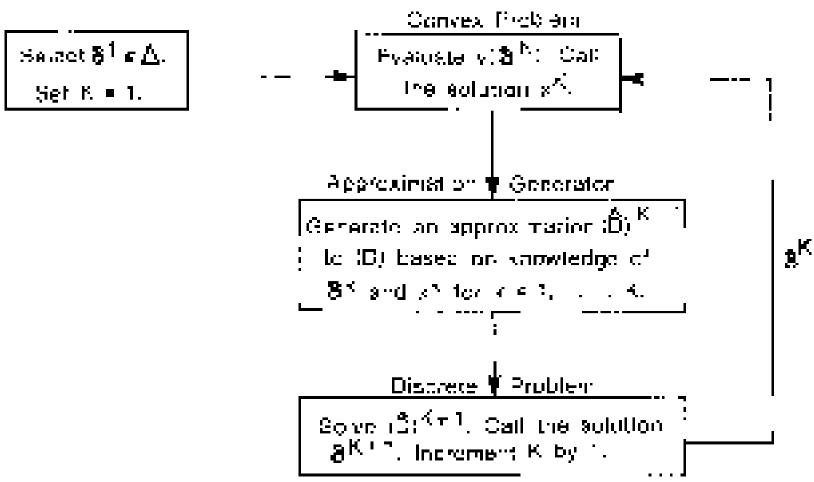


Fig. 2.

The principal varieties of this approach are determined by the type of rule used to construct $(\hat{D})^K$. The most widely known rule is probably the one specified by Benders decomposition for the case in which X_δ can be written as

$$X_\delta = \{x \in X : G_\delta(x) \leq 0\}, \tag{8}$$

where X is a convex set independent of δ and, for each δ, G_δ is a vector of convex functions. Benders' rule specifies a global lower approximation to $v(\cdot)$ which, at the Kth step, is the upper envelope of all lower approximations to $v(\cdot)$ generated at previous steps as a byproduct of evaluating $v(\delta^k)$.

Benders decomposition is well known in the context of mixed integer linear programming and need not be described in detail here [15, 22]. The most appropriate version in the context of (DC) is a generalization worked out by the author elsewhere [17] which avoids having to assume: a) that $f_\delta(\cdot)$ and $G_\delta(\cdot)$ in (8) are additively separable in x and δ and linear in x, and b) that X is the nonnegative orthant. The generalization does, however, require a certain mathematical property to hold in order for the computational procedure to be practical (see [17, p. 251]). In any case, an examination of the essential arguments of [17] shows that the basic finite convergence theorem (Th. 2.4) holds whether or not Δ is a subset of a vector space.

See [20] for a detailed description of a successful application of Benders decomposition to a multicommodity distribution system design problem. It combines a specialized pure 0–1 integer programming algorithm with an algorithm for the classical transportation problem.

A completely different class of rules for constructing $(\hat{D})^K$ is obtained by introducing the notion of a *policy function* $p(\cdot)$ which associates a point in X_δ with every δ in Δ. The ideal policy function $p^*(\cdot)$ obviously is one which specifies the minimizing value of x for f_δ over X_δ as a function of δ, in which case one has

$$f_\delta(p^*(\delta)) = v(\delta) \quad \text{and } p^*(\delta) \in X_\delta \quad \text{for all } \delta \in \Delta. \tag{9}$$

Situations where f_δ does not achieve its infimum over X_δ, or where X_δ is empty, could be accommodated by standard devices but will not be discussed here. The job of the Approximation Generator (See Fig. 2) at the Kth iteration is to specify the next approximation $p^{K+1}(\cdot)$ to $p^*(\cdot)$ which takes advantage of the previous information obtained via Property 1 and yet leads to a mathematical structure of

$$\min_{\delta \in \Delta} c_\delta + f_\delta(p^{K+1}(\delta)) \tag*{$(\hat{D})^{K+1}$}$$

which is tractable within the scope of Property 2. Within whatever latitude may be offered by Property 2, it seems desirable to require

$$p^{K+1}(\delta^k) = x^k \quad \text{for } k = 1, \ldots, K \tag{10}$$

and that $p^{K+1}(\cdot)$ otherwise be as simple a function as is consistent with any known properties of $p^*(\cdot)$, which in turn depend intimately on the structure of f_δ, X_δ, and

Δ. Finite termination follows trivially from the finiteness of S if (10) can be enforced with exact equality for all K.

Except for the trivial case where $p^{K-1}(\cdot)$ is taken as identically equal to x^K, I am unable to cite an instance where the policy approximation approach has been used. This presents an attractive research opportunity.

An interesting comparison can be drawn between the Benders decomposition approach and the policy approximation approach: the former approximates $v(\cdot)$ from below by constructs in the range space of f_1 and G_1, while the latter approximates $v(\cdot)$ from above by constructs in the domain space of f_1.

4. Conclusions and opportunities for research

We have defined a category of optimization problems, herein dubbed discrete/convex programs, which has numerous practical applications and also lends itself to the development of hybrid algorithms that exploit the individual tractability of the discrete and convex aspects of the problem taken separately (so-called Properties 1 and 2).

Much work remains to be done before discrete/convex programming reaches maturity in its ability to synthesize practical hybrid algorithms from the separate algorithmic repertoires of discrete optimization and linear or convex programming. One important task is to accumulate a broader and more detailed inventory of applicable discrete/convex models along with specific statements of Properties 1 and 2 for each. This is necessary in order to discern the practical scope of the field more clearly and to provide grist for the mill of hybrid algorithmic development.

Another important undertaking is to study the computational approaches outlined here in more detail, both individually and with reference to similar hybrid algorithms already available in the literature. Among the interesting questions for study in various applications contexts are the following:

For the combinatorial seeding with local convex enumeration approach:

How does the quality of the "seed" interact with the definition of the enumeration "neighborhood" to determine the total enumerative work and the degree of global optimality upon termination?

Do some neighborhood definitions facilitate particularly efficient implicit enumeration techniques in the local convex enumeration phase?

For the generalized branch-and-bound approach:

What useful kinds of relaxation or other bound-producing operations exist in addition to convexification and Lagrangean relaxation?

What are the most effective ways to determine the best multipliers for generalized Lagrangean relaxation?

To what extent do the accumulated auxiliary devices and conventional wisdom of integer linear programming carry over to the more general context of (DC)?

For the cyclic marginal optimization approach:

Under what conditions does marginal optimality imply global optimality?

For the improving approximations approach to (D):

Are there applications where (8) does not hold and yet approximation rules can be devised with properties similar to those of Benders decomposition?

Are there applications where useful properties of $\rho^*(\cdot)$ can be derived to guide the policy approximation approach? (Properties of optimal policies have been a traditional concern in dynamic programming and inventory theory, but have yet to receive serious attention in modern computationally oriented mathematical programming.)

What other promising kinds of approximation generators for (D) are there besides the two types discussed herein?

Finally, what other attractive approaches exist besides the four discussed in this paper for the development of hybrid algorithms for discrete/convex programming?

References

[1] G. Bergendahl, *Models for investments in a road network*, Department of Business Administration, Stockholm University, Monograph No. 1, Bonniers, 1969.

[2] D.E. Boyce, A. Farhi and R. Weischedel, Optimal network problem: A branch-and-bound algorithm, *Environment and Planning*, 5 (1973) 519–533.

[3] W. Chou, Computer communication networks — the parts make up the whole, 1975 National Computer Conference, *AFIPS Conference Proceedings* (AFIPS Press, Montvale, NJ, 1975) 119–128.

[4] W. Chou and A. Kershenbaum, A unified algorithm for designing multidrop teleprocessing networks, *Proc. Third IEEE Symposium on Data Networks Analysis and Design*, St. Petersburg, Florida, November 13–15, 1973, pp. 148–156.

[5] N. Christofides and P. Brooker, Optimal expansion of an existing network, *Math. Programming*, 6 (1974) 197–211.

[6] J.E. Day and M.P. Hottenstein, Review of sequencing research, *Naval Res. Logist. Quart.*, 17 (1970) 11–39.

[7] S.E. Elmaghraby, The machine sequencing problem — review and extensions, *Naval Res. Logist. Quart.*, 15, 2 (June 1968) 205–232.

[8] D. Erlenkotter and J. Scott Rogers, Sequencing competitive expansion projects, Working Paper No. 234, Western Management Science Institute, UCLA, June 1975.

[9] M. Florian and S. Nguyen, A method for computing network equilibrium with elastic demands, Publication #2, Centre de Recherche Sur Les Transports, University of Montreal, January 1974.

[10] R.L. Francis and J.A. White, *Facilities Layout and Location: An Analytical Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1974).

[11] H. Frank, I.T. Frisch and W. Chou, Topological considerations in the design of the ARPA computer network, SJCC 1970, *AFIPS Conference Proceedings* (AFIPS Press, Montvale, NJ) pp. 581–587.

[12] H. Frank, R.E. Kahn and L. Kleinrock, Computer communication network design — experience with theory and practice, SJCC 1972, *AFIPS Conference Proceedings* (AFIPS Press, Montvale, NJ) pp. 255–270.

[13] L. Fratta, M. Gerla and L. Kleinrock, The flow deviation method: An approach to store-and-forward communications network design, *Networks*, 3 (1973) 97–133.

[14] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming* (Wiley, New York, 1972).

[15] Geoffrion, A.M., Elements of large-scale mathematical programming, *Management Sci.* 16 (1970) 652–691.

[16] A.M. Geoffrion, Duality in nonlinear programming, *SIAM Review*, 13 (1971) 1–37.

[17] A.M. Geoffrion, Generalized Benders decomposition, *J. Optimization Theory and Appl.* 10 (1972) 237–260.

[18] A.M. Geoffrion, Lagrangean relaxation for integer programming, *Math. Programming Study* 2 (1974) 82–114.

[19] A.M. Geoffrion, A guide to computer-assisted methods for distribution systems planning, *Sloan Management Review*, 16 (1975) 17–41.

[20] A.M. Geoffrion and G.W. Graves, Multicommodity distribution system design by Benders decomposition, *Management Sci.*, 20 (1974) 822–844.

[21] A.M. Geoffrion and G.W. Graves, Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/LP approach, *Operations Res.* 24 (1976) 595–610.

[22] A.M. Geoffrion and R.E. Marsten, Integer programming algorithms: A framework and state-of-the-art survey, *Management Sci.*, 18 (1972) 465–491.

[23] F.S. Hillier and M.M. Connors, Quadratic assignment problem algorithms and the location of indivisible facilities, *Management Sci.*, 13 (1966) 42–57.

[24] M.L. Hong, Experiments with a job shop production model coupling a linear program and a heuristic scheduling procedure, M.S. Thesis, Graduate School of Management, UCLA, 1974.

[25] A.C. Lea, Location-allocation systems: An annotated bibliography, Discussion Paper No. 13, Dept. of Geography, University of Toronto, May 1973.

[26] T.L. Magnanti, J.F. Shapiro and M.H. Wagner, Generalized Linear Programming solves the Dual, *Management Sci.* 22 (1976) 1195–1203.

[27] H. Muller-Merbach, Heuristic methods: Structures, applications, computational experience, in R.W. Cottle and J. Krarup, eds., *Optimization Methods* (English Universities Press, London, 1974).

[28] T. Prabhakar, Some scheduling applications in the chemical industry, in *Symp. Theory of Scheduling and its Appl.*, 86 (Springer, Berlin, 1973).

[29] T. Prabhakar, A production scheduling problem with sequencing considerations, *Management Sci.*, 21 (1974) 34–42.

[30] R.T. Richardson, Benders' decomposition method applied to an airline routing problem, Ph.D. Dissertation, Dept. of Industrial Engineering, Systems Management Engineering and Operations Research, University of Pittsburgh, 1975.

[31] G.T. Ross and R.M. Soland, A branch and bound algorithm for the generalized assignment problem, *Math. Programming*, 8 (1975) 91–103.

[32] B. Rothfarb et al., Optimal design of offshore natural gas pipeline systems, *Operations Res.*, 18 (1970) 992–1020.

[33] E.R. Ruiter, The prediction of network equilibrium — the state of the art, Transportation Systems Division, Dept. of Civil Engineering, MIT, June 1973.

[34] C. Toregas, R. Swain, C. ReVelle and L. Bergman, The location of emergency service facilities, *Operations Res.*, 19 (1971) 1363–1373.

[35] W.I. Zangwill, *Nonlinear Programming* (Prentice-Hall, Englewood Cliffs, NJ, 1969).

Annals of Discrete Mathematics 1 (1977) 221-231

ON INTEGER AND MIXED INTEGER FRACTIONAL PROGRAMMING PROBLEMS*

Daniel GRANOT

Department of Economics and Commerce and Computing Science Program, Simon Fraser University, Burnaby, BC, Canada

Frieda GRANOT

Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, BC, Canada

We construct in this paper new cutting plane algorithms for solving the Integer Fractional Programming (IFP) and the Mixed Integer Fractional Programming (MIFP) problems.

By using Charnes and Cooper's approach for solving continuous fractional programs we develop two types of cutting planes, which can be systematically generated and applied while solving (IFP) problems. Similar results are obtained for the (MIFP) problem.

By employing Martos' approach for solving continuous fractional programs together with Young's primal algorithm for solving Integer Programming problems, we are able to construct a primal algorithm for solving (IFP) problems in finitely many iterations.

1. Introduction

The Integer Fractional Programming (IFP) problem can be formulated as:

$$\text{(IFP):} \qquad \max\{(c^T x + c_0)/(d^T x + d_0)\}$$

$$\text{s.t. } Ax \leq b, \quad x \geq 0, \quad x \text{ integer.}$$

Problems with linear fractional objective function arise, e.g., in attrition games [13], Markovian replacement problems [5, 14], the cutting stock problem [7], primal dual approaches to decomposition procedures [2, 15], and portfolio theory [19, 23]. If the variables in the fractional model represent indivisible commodities, then restricting them to integer values results with the (IFP) formulation. For example, in [21] the Mining for Investment Return problem was formulated as an (IFP) problem.

Robillard [18] has developed an algorithm for solving a special class of {0, 1} fractional programs. The algorithms developed by Florian and Robillard [6], Grunspan and Thomas [12] and Anzai [1] for solving (IFP) problems are based on Isbell and Marlow's results for continuous fractional programs [13]. Their algorithms reduce the problem of solving (IFP) to that of solving a finite sequence of linear integer programming problems.

* This research was partly supported by NRC Grants Number A—3024, A—[illegible] and A-4181.

In this paper we construct new algorithms for solving (IFP) and Mixed Integer Fractional Programming (MIFP) problems. In contrast with the results in [1, 6, 12], which are based on Isbell and Marlow's approach to solve fractional programs, our algorithms are based on Charnes and Cooper's method [4] and on Martos' method [17] for solving continuous fractional programs. More specifically, applying Charnes and Cooper's transformation [4] on (IFP) results with an equivalent problem, denoted by (IFP1). By exploiting the relationship between (IFP) and (IFP1) we develop two types of cutting planes which can be systematically generated and applied while solving (IFP) problems. Similar results are obtained for the (MIFP) problem. Also, based on Martos' [17] and on Young [22], or Glover [8], a primal algorithm for solving the (IFP) problem in finitely many iterations is developed.

2. Cut A for (IFP)

Consider again the (IFP) problem:

$$\max[(c^T x + c_0)/(d^T x + d_0)] \tag{1}$$

$$\text{s.t. } Ax \leq b \tag{2}$$

$$x \geq 0, \quad x \text{ integer} \tag{3}$$

where A is an $m \times n$ matrix, c^T, d^T and b^T are given row vectors, c_0 and d_0 are scalars and x is an $n \times 1$ column vector of unknown variables.

Let us denote by

$$S = \{x : Ax \leq b, x \geq 0\} \tag{4}$$

and assume that

Assumption 1: $d^T x + d_0 > 0$ on S.

Assumption 2: S is a non-empty and bounded set in $\mathbf{R}^n$.

The difficulty in solving (IFP) problems stems from the fact that the algorithms for solving continuous fractional programs, in which the objective function is maintained in its original form (1), require that primal feasibility will be satisfied in each iteration. Therefore, one cannot hope to solve an (IFP) problem by applying a dual cutting plane algorithm, e.g., that in [9], directly on (1), (2), (3). In order to circumvent this difficulty we shall first apply Charnes and Cooper's transformation [4] on (IFP) to obtain an equivalent problem, denoted by (IFP1), of the form:

$$\max[z = c^T y + c_0 t] \tag{5}$$

$$\text{s.t. } Ay - bt \leq 0 \tag{6}$$

$$d^T y + d_0 t = 1 \tag{7}$$

$$y, t \geq 0, \quad y/t \text{ integer} \tag{8}$$

where $y = tx$. Then, we shall construct cutting-planes which can be used for solving the equivalent (IFP1) problem.

Remark 1. Assumption 1 above is not restrictive in the sense that when not satisfied we may have to solve two or at most three problems of the form (IFP1). Assumption 2 implies that $t > 0$ in every feasible solution (y, t) to (IFP1), see [4].

Theorem 1. *If $d^T x^* + d_0 > 0$ for x^*, an optimal solution for* (IFP), *and if (y^*, t^*) is an optimal solution for* (IFP1), *then y^*/t^* is an optimal solution for* (IFP).

Proof. Similar to that of [4, Theorem 1], hence omitted.

From Theorem 1 we conclude that in order to solve (IFP) it is sufficient to solve the equivalent problem (IFP1).

Let us solve the (LP) problem associated with (IFP1), after introducing slack variables to convert inequalities in (6) to equalities. Then z and the basic variables in the optimal tableau can be expressed in terms of the non-basic variables as follows:

$$z = a_{00} + \sum_{j \in I_N} \bar{a}_{0j}(-y_j), \tag{9}$$

$$y_i = a_{i0} + \sum_{j \in I_N} \bar{a}_{ij}(-y_j), \qquad i \in I_B, \tag{10}$$

$$t = a_{m+1,0} + \sum_{j \in I_N} \bar{a}_{m+1,j}(-y_j), \tag{11}$$

where I_B, I_N are the set of indices corresponding to the basic variables (excluding t) and the non-basic variables, respectively.

Note that Assumption 2 implies that t is a basic variable at the optimal solution. Clearly, from the optimality criterion, $\bar{a}_{0j} \geq 0$, $j \in I_N$. Now if $a_{i0}/a_{m+1,0}$ is integer for all $i \in I_B$ then an optimal solution x^* to (IFP) is given by $(x_i^* = a_{i0}/a_{m+1,0}$, $i \in I_B$, $x_i^* = 0$, $i \in I_N)$. Otherwise, there exists at least one index, say k, for which $a_{k0}/a_{m+1,0}$ is not integer.

Naturally, when striving to satisfy the integrality restriction, one is tempted to use the k^{th} basic constraint as a source row for generating a Gomory's cut. However, this might be somewhat complicated due to the congruence relation, $y/t \equiv 0$ modulo 1, in (8). In order to overcome this difficulty we shall resort to the relationship between (IFP) and (IFP1), which was established for continuous fractional programs in [4, 29].

Let us denote by $\bar{B}$ the optimal basis associated with the current optimal continuous solution to (IFP1). Since t is a basic variable, $\bar{B}$ can be partitioned into

$$\bar{B} = \begin{pmatrix} B & -b \\ d_B & d_0 \end{pmatrix}$$

where d_B contains the components of d corresponding to B. Further, it can be shown by matrix calculation, see also [20], that if $\hat{B}^{-1}$ is partitioned into

$$\hat{B}^{-1} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

where $M_{11} \in R^{m \times m}$, then

$$M_{11} = B^{-1} - B^{-1}b(d_0 + d_B B^{-1}b)^{-1} d_B B^{-1}, \qquad M_{21} = -(d_0 + d_B B^{-1}b)^{-1} d_B B^{-1},$$

$$M_{12} = (d_0 + d_B B^{-1}b)^{-1} B^{-1}b, \qquad M_{22} = (d_0 + d_B B^{-1}b)^{-1},$$

and therefore B^{-1} can be determined by:

$$B^{-1} = M_{11} - B^{-1}bM_{21} = M_{11} - x_B M_{21} \tag{12}$$

Note that Assumption 1 implies that $d_0 + d_B B^{-1}b > 0$.

Using (2) and the relationship between (IFP) and (IFP1), the continuous fractional problem associated with (IFP) (after adding slack variables) can be equivalently written as:

$$\max \left\{ \left(\bar{c}_0 + \sum_{j \in J_N} \bar{c}_j x_j \right) \Big/ \left(\bar{d}_0 - \sum_{j \in J_N} \bar{d}_j x_j \right) \right\} \tag{13}$$

$$\text{s.t. } x_i = a_{i0}/a_{m+1,0} + \sum_{j \in J_N} \bar{a}_{ij}(-x_j) \quad i \in I_B,$$
$$x \geq 0, \quad x \text{ integer} \tag{14}$$

where $\bar{a}_j = (B^{-1})_i N_j$ and N_j is the column of A corresponding to the non-basic variable x_j.

By assumption, $a_{k0}/a_{m+1,0}$ is not integer and therefore the kth constraint in (14) can serve as a source row for generating a Gomory's cut, see [10], of the form

$$\sum_{j \in J_N} \bar{f}_{kj} x_j \geq \bar{f}_{k0} \tag{15}$$

where

$$0 \leq \bar{f}_{kj} = \bar{a}_{kj} - [\bar{a}_{kj}] < 1, \qquad 0 < \bar{f}_{k0} = a_{k0}/a_{m+1,0} - [a_{k0}/a_{m+1,0}] < 1$$

and $[\alpha]$ denotes the largest integer smaller than or equal to α. Inequality (15) should be satisfied by any feasible solution to (IFP). Multiplying (15) from both sides by t, $t > 0$, substituting $tx_j = y_j$, $j \in J_N$, and using (11) to express t in (15), in terms of the non-basic variable results with the constraint

$$s = -\bar{f}_{k0}\, a_{m+1,0} + \sum_{j \in J_N} (-\bar{f}_{kj} - \bar{f}_{k0}\bar{a}_{m+1,j})(-y_j) \geq 0, \quad s \equiv 0 \text{ modulo } t. \tag{16}$$

Clearly, (16) is not satisfied by the current continuous optimal solution represented by (10), (11). Thus, whenever a constraint of the form (16) is appended to the optimal tableau it cuts off the optimal continuous solution but not any integer feasible solution for (IFP). Cut of the for (16), to which we shall refer to as Cut A can be systematically generated and appended to (IFP1) whenever the continuous optimal solution does not satisfy the integrality requirements.

We remark that other cuts which were offered to Integer Programs can be used, in a similar manner, to generate cuts for (IFP) problems, e.g., Martin's "accelerated" cut [16]

3. Cut B for (IFP)

By using similar arguments to those used by Gomory [9], we are able to construct another cutting plane which can be systematically generated and employed when solving (IFP) problems. In contrast with Cut A, the cut to be constructed in this section, to which we shall refer to as Cut B, is generated directly from (IFP1). However, while Cut A can be applied whenever the optimal solution to the associated continuous problem does not satisfy the integrality requirement, Cut B can be applied only when an additional requirement, which can be easily verified at the outset, is met.

Let us consider again the (IFP1) problem and let $\bar{t}$ be a lower bound for t in (IFP1). Such a value can always be secured by solving the (LP) problem

$$\max\{d^T x + d_0 \quad \text{s.t.} \quad Ax \le b,\ x \ge 0\}, \tag{17}$$

and taking $\bar{t} = 1/(d^T x^* + d_0)$ where x^* is an optimal solution for (17). Assumptions 1, 2 guarantee that $\bar{t} > 0$.

Let us assume again that $a_{k0}/a_{m+1,0}$ in (10), (11) is not integer, and consider the following two equations taken from (10), (11)

$$y_k = a_{k0} + \sum_{j \in J_N} \hat{a}_{kj}(-y_j), \tag{18}$$

$$t = a_{m+1,0} + \sum_{j \in J_N} \hat{a}_{m+1,j}(-y_j). \tag{19}$$

From $y_k \equiv 0$ modulo t we have

$$a_{k0} + \sum_{j \in J_N} \hat{a}_{kj}(-y_j) \equiv 0 \quad \text{modulo } t. \tag{20}$$

Further, since the value of t is always given by (19), we can add or subtract (19) from (20) as many times as necessary in order to obtain

$$f_{k0} + \sum_{j \in J_N} \bar{f}_{kj}(-y_j) \equiv 0 \quad \text{modulo } t, \tag{21}$$

where

$$0 < f_{k0} = a_{k0} - [a_{k0}/a_{m+1,0}]\,a_{m+1,0}, \quad \bar{f}_{kj} = \hat{a}_{kj} - [a_{k0}/a_{m+1,0}] \cdot \hat{a}_{m+1,j},$$

Moreover, we can use the relations

$$y_j \equiv 0 \text{ modulo } t, \quad j \in I_N \tag{22}$$

to obtain

$$\sum_{j \in I_N} f_{kj} y_j \equiv f_{k0} \text{ modulo } t \tag{23}$$

where

$$0 \leqslant f_{kj} = \bar{f}_{kj} - [\bar{f}_{kj}] < 1.$$

From (23) we conclude that either

$$\sum_{j \in I_N} f_{kj} y_j = f_{k0}, f_{k0} + t, f_{k0} + 2t, \dots \tag{24}$$

or

$$\sum_{j \in I_N} f_{kj} y_j = f_{k0} - t, f_{k0} - 2t, \dots . \tag{25}$$

However, $f_{kj} \geqslant 0$ and $y_j \geqslant 0\ \forall j \in I_N$, thus if $f_{k0} < t$ only relation (24) is feasible and can then be replaced by the constraint

$$s_1 = -f_{k0} + \sum_{j \in I_N} f_{kj} y_j \geqslant 0, \qquad s_1 \equiv 0 \text{ modulo } t \tag{26}$$

which should be satisfied by an optimal solution to (IFP1). Clearly, (26) is not satisfied by the current optimal solution to (IFP1). Therefore, whenever there exists y_k for which $a_{k0}/a_{r+1,k}$ is not integer and $a_{k0} - [a_{k0}/a_{r+1,k}] \cdot a_{r+1,k} < t$, a cut of the form (26) can be appended to (10), (11) which will cut off the non-integer optimal solution to (IFP1).

4. A primal algorithm for integer fractional programs

In this section a primal all integer algorithm for solving (IFP) is presented. The algorithm proceeds to an optimal solution for (IFP) through a finite sequence of feasible solutions. It is applied directly to (IFP) in a format originally suggested by Martos [17] for continuous fractional programs, and is a direct and natural extension of the primal algorithm for linear integer programs, see e.g., [2, 8, 22], to (IFP) problems.

Consider again the (IFP) problem in which inequality constraints were converted to equalities by introducing slack variables. Assume that all the given data in (IFP) is in integers and that a feasible integer solution to (IFP) is at hand. Thus, (IFP) can be equivalently written as:

$$\max \left\{ \left(\bar{c}_0 + \sum_{j \in I_N} \bar{c}_j x_j \right) \Big/ \left(\bar{d}_0 + \sum_{j \in I_N} \bar{d}_j x_j \right) \right\} \quad \text{s.t.} \quad I x_B + A x_N = a_0, \tag{27}$$

$$x_B, x_N \geqslant 0 \text{ and integers.}$$

where $\bar{a}_0 \geq 0$ and integer, $\bar{A}$ is a matrix of integer entries. x_B and x_N are vectors of basic and non-basic components, respectively, and I_N is the set of indices corresponding to non-basic variables.

Clearly, neither Cut A nor B can be employed in a primal algorithm for solving (IFP) problems, since adding any of these cuts to the constraints of (IFP) will destroy primal feasibility. The primal algorithm to be presented in this section is based on Martos' [17] adjacent extreme point algorithm for solving continuous fractional programs. In Martos' algorithm the original structure of the constraints is maintained, and the iterations are carried out in an augmented simplex tableau which includes $m + 3$ rows. The first m rows correspond to the original constraints, the $m + 1$ and $m + 2$ rows correspond to the numerator and denominator of the fractional function, respectively, and the last row corresponds to the $\bar{r}_j$'s where

$$\bar{r}_j = \bar{c}_0 \bar{d}_j - \bar{d}_0 \bar{c}_j, \quad j \in I_N. \tag{28}$$

In every iteration of the algorithm the first $m + 2$ rows are modified through the ordinary pivot operations, whereas the last row is modified via (28).

Now, if $\bar{r}_j \leq 0$, $j \in I_N$, in (27), then $(x_B = \bar{a}_0, x_N = 0)$ is an optimal solution to (IFP). Otherwise, there exists an index k, $k \in I_N$, for which $\bar{r}_k > 0$. Let

$$\theta_k = \min\{\bar{a}_{i0}/\bar{a}_{ik} : \bar{a}_{ik} > 0\}. \tag{29}$$

Then any row u for which $\lfloor \bar{a}_{u0}/\bar{a}_{uk} \rfloor \leq \theta_k$ can serve as a source row for generating a Gomory's cut of the form

$$s + \sum_{j \in I_N} \lfloor \bar{a}_{uj}/\bar{a}_{uk} \rfloor x_j = \lfloor \bar{a}_{u0}/\bar{a}_{uk} \rfloor, \quad s \geq 0 \tag{30}$$

This cut was first suggested by Gomory in [10] for his all integer algorithm, and was used subsequently by Ben Israel and Charnes [3] to construct their all integer primal algorithm for (IP).

In order to solve (IFP), cut (30) can be added to (27) and serve as a pivot row, with the k^{th} column as a pivot column. Since the value of the pivot in this case is $\lfloor \bar{a}_{uk}/\bar{a}_{uk} \rfloor = 1$, the new coefficients obtained after performing the ordinary pivot operations are all integers. Moreover, adding (30) to (27) does not exclude any feasible integer solution to (27). The slack variable s in (30) will be a new basic variable whose value in the new tableau will be $\lfloor \bar{a}_{u0}/\bar{a}_{uk} \rfloor$.

Whenever $\lfloor \bar{a}_{u0}/\bar{a}_{uk} \rfloor = 0$ a stationary cycle[1] occurs, and the value of the constant vector is not changed. Since we assumed that $S = \{x : Ax \leq b, x \geq 0\}$ is bounded, a primal algorithm for (IFP) will converge in a finite number of iterations if we can guarantee that any sequence of stationary cycles is finite[2]. In the (IFP) problem,

[1] The problem of finiteness in the primal algorithm for (IP) is sometimes clarified by the distinction between stationary cycles and transition cycles. A stationary cycle is a degenerate cycle in which $\lfloor \bar{a}_{u0}/\bar{a}_{uk} \rfloor = 0$, whereas in a transition cycle the objective function is strictly increased.

[2] For a very thorough discussion of the problem of finiteness in the primal algorithm for (IP) via the distinction between stationary and transition cycles the reader is referred to [22].

since the last row is modified via (28), we cannot establish strict lexicographical decrease of a certain column vector, the way it was done in [8] or [22]. Thus, a finiteness proof of a primal algorithm for (IFP) problems, in which we systematically generate cuts of the form (30), is not available at this stage.

Let us superscript the elements obtained from (27), (30) after performing one pivot iteration by (^). Then,

$$\hat{a}_0 = \bar{a}_0 - [\bar{a}_{u0}/\bar{a}_{uk}]\bar{a}_k \tag{31}$$

where

$$a_k = (\bar{a}_{1k}, \bar{a}_{2k}, \ldots, \bar{a}_{mk})^T,$$

$$\begin{aligned}\hat{t}_j = \hat{c}_0\hat{d}_j - \hat{d}_0\hat{c}_j &= (\bar{c}_0 - \bar{c}_k[\bar{a}_{u0}/\bar{a}_{uk}])(\bar{d}_j - \bar{d}_k[\bar{a}_{uj}/\bar{a}_{uk}]) \\ &\quad - (\bar{d}_0 - \bar{d}_k[\bar{a}_{u0}/\bar{a}_{uk}])(\bar{c}_j - \bar{c}_k[\bar{a}_{uj}/\bar{a}_{uk}]) \\ &= (\bar{c}_0\bar{d}_j - \bar{d}_0\bar{c}_j) - [\bar{a}_{u0}/\bar{a}_{uk}](\bar{c}_k\bar{d}_j - \bar{d}_k\bar{c}_j) - [\bar{a}_{uj}/\bar{a}_{uk}](\bar{c}_0\bar{d}_k - \bar{d}_0\bar{c}_k) \\ &= \bar{t}_j - [\bar{a}_{u0}/\bar{a}_{uk}]t_{kj} - [\bar{a}_{uj}/\bar{a}_{uk}](\bar{c}_0\bar{d}_k - \bar{d}_0\bar{c}_k),\end{aligned}$$

where k is the pivot column and u is the source row in (30).

In a stationary cycle $[\bar{a}_{u0}/\bar{a}_{uk}] = 0$ and thus, for a stationary cycle

$$\hat{t}_j = \bar{t}_j - [\bar{a}_{uj}/\bar{a}_{uk}]\bar{t}_k. \tag{32}$$

Therefore, the modification of the last row via (32) in stationary cycles can simply be achieved through the ordinary pivot operations rather than by (28). Moreover, (32) indicates that in stationary cycles the linear fractional objective function can be replaced, for tableau modification sake, by a linear objective function whose relative cost coefficients are the $\bar{t}_j$'s.

The above observation in conjunction with Young's ingenious reference row [22] (see also Glover [5]) can be used to construct a primal algorithm, in which cut (30) is systematically generated whenever for some $k \in I_N$, $\bar{t}_k > 0$, which converges to an optimal solution to (IFP) in finitely many iterations.

5. Mixed integer fractional programming (MIFP)

The mixed integer fractional programming problem is an optimization problem of the form (MIFP):

$$\begin{aligned}&\max\{(c_1^T x + c_2^T y + c_0)/(d_1^T x + d_2^T y + d_0)\}, \\ &\text{s.t. } A_1 x + A_2 y \leq b, \\ &\qquad x, y \geq 0, \quad x \text{ integer.}\end{aligned}$$

Let us denote by

$$S = \{(x, y) : A_1 x + A_2 y \leq b, \ x, y \geq 0\},$$

and assume

Proof. Clearly (6) is equivalent to (2) if $m = n$. We have to prove that (6) implies $(r+2)$-connectedness.

If there is a k with $r < k < \frac{1}{2}(n+r)$ such that $d_{k-r} \le k$, then by the arguments of the proof of Theorem 8, Section (I) (a) $(r+2)$-connectedness is assured.

If $d_{k-r} > k$ for all $r < k < \frac{1}{2}(n+r)$, we have $d_q \ge q + r$, where $q := \left\lceil \frac{n-r}{2} \right\rceil$. Furthermore $2q \ge n - r$ and $q \le n - r - 1$ (as $r \le n - 3$), thus $q + r \le d_q \le d_{q+r}$. This implies

$$q = 2q - q \ge n - (r + q) > n - (q + r) - 1 \ge n - d_{q+r} - 1.$$

Thus condition (1) of Proposition 1 is satisfied and G is $(r+2)$-connected. □

Actually Berge proved a stronger theorem saying that Q only has to be a set of edges of cardinality r such that the connected components of Q are paths.

Corollary 13 (Chvátal [4]). *If the degree sequence $d_1, \ldots, d_n$ of a graph G, $n \ge 3$, satisfies*

$$d_k \le k < \tfrac{1}{2}n \implies d_{n-k} \ge n - k, \tag{7}$$

then G contains a hamiltonian cycle.

Proof. Take $r = 0$ in Corollary 12. □

Furthermore, Chvátal showed that this theorem is best possible in the sense that if there is a degree sequence of a graph not satisfying (7) then there exists a non-hamiltonian graph having a degree sequence which majorizes the given one. This proves that Theorem 8 is also best possible in this special case. Moreover Chvátal (see [4]) showed that most of the classical results on hamiltonian graphs are contained in his theorem, and therefore are also implied by Theorem 8.

A trivial consequence of Corollary 13 which however is not too "workable" is

Corollary 14. *Let G' be an induced subgraph of a graph G having $m \le n$ vertices. If the degree sequence $d'_1, \ldots, d'_m$ of G' satisfies (7) then G contains a cycle of length m.* □

4. Some examples

(a) We first show that the number m implied by Theorem 8 giving the minimum length of a cycle containing a given path cannot be increased, i.e. we give an example of a graph G with a path Q of length r such that the longest cycle containing Q has length m.

Consider a graph with two disjoint vertex sets A and B. A is a clique of q

where

$$I_k^{i+} = \{j : j \in I_k^i,\ \hat a_{kj} \geq 0\},\quad I_k^{i-} = \{j : j \in I_k^i,\ \hat a_{kj} < 0\} \quad (i = 1, 2).$$

If $\bar t$ is a lower bound for t in (MIFP) then from assumptions 1', 2' we conclude that $\bar t > 0$. Further, since for every feasible z, $t \geq \bar t$ (37) implies

$$\sum_{j \in I_k^{2+}} \hat a_{kj} z_j^1 + \sum_{j \in I_k^{2-}} \hat a_{kj} z_j^2 \geq f_{k0} - \bar t. \tag{38}$$

Moreover, if $f_{k0} - \bar t < 0$ then by multiplying (38) from both sides by $f_{k0}/(f_{k0} - \bar t) < 0$ we obtain

$$\sum_{j \in I_k^{2+}} (\hat a_{kj} f_{k0}/(f_{k0} - \bar t)) z_j^1 - \sum_{j \in I_k^{2-}} (\hat a_{kj} f_{k0}/(f_{k0} - \bar t)) z_j^2 \geq f_{k0}. \tag{39}$$

Combining now inequalities (36) and (39) results with

$$\sum_{j \in I_k^{1+}} \hat a_{kj} z_j^1 - \sum_{j \in I_k^{1-}} \hat a_{kj} z_j^2 + \sum_{j \in I_k^{2+}} (\hat a_{kj} f_{k0}/(f_{k0} - \bar t)) z_j^1 +$$
$$- \sum_{j \in I_k^{2-}} (\hat a_{kj} f_{k0}/(f_{k0} - \bar t)) z_j^2 \geq f_{k0}. \tag{40}$$

Inequality (40) is not satisfied by the current optimal solution to $(\overline{\text{MIFP}})$, and when added to the bottom of the optimal tableau it cuts off the optimal continuous solution to (MIFP). Further, by using the fact that $z_j \equiv 0$ modulo λ in (40) one can obtain the following stronger cut

$$\sum_{j \in I_k} f_{kj}^{*} z_j^1 + \sum_{j \in I_k} f_{kj}^{*} z_j^2 \geq f_{k0}, \tag{41}$$

where

$$f_{kj}^{*} = \begin{cases} \hat a_{kj} & \text{if } j \in I_k^{1+}, \\ f_{kj} & \text{if } j \in (I_k^{1+} \cup I_k^{1-}) \text{ and } \bar t \cdot f_{kj} \leq f_{k0}, \\ f_{k0}(1 - f_{kj})/(\lambda - f_{k0}) & \text{if } j \in (I_k^{1+} \cup I_k^{1-}) \text{ and } \bar t \cdot f_{kj} > f_{k0}, \\ \hat a_{kj} f_{k0}/(\lambda - f_{k0}) & \text{if } j \in I_k^{2}. \end{cases} \tag{42}$$

Thus, if for some variable z_k, $a_{k0}/a_{m+1,0}$ is not integer, cut (41) can be applied while solving (MIFP), provided $\bar t$ (the lower bound for t) satisfies

$$a_{k0} - [a_{k0}/a_{m+1,0}] \cdot a_{m+1,0} \leq \bar t. \tag{43}$$

One can show that if $I_k^2 = \emptyset$, i.e., all variables in (33), (34) are constrained to be congruent to zero modulo λ and if $\{j : \bar t \cdot f_{kj} > f_{k0}\} \neq \emptyset$ then cut (41) is stronger than Cut A which was derived in section 2 for the (IFP) problem.

References

[1] Y. Anzai, On integer fractional programming, *J. Operations Res. Soc. Japan* 17 (1974) 49–66.

[2] E.J. Bell, Primal-dual decomposition programming, Ph.D. Thesis, Operations Research Center, University of California at Berkeley, Report ORC 65-92, (1965).

[3] A. Ben-Israel and A. Charnes, On some problems of diophantine programming, *Cahiers Centre Études Recherche Opér.* 4 (1962) 215–280.

[4] A. Charnes and W.W. Cooper, Programming with linear fractional functionals, *Naval Res. Logist. Quart.* 9 (1962) 181–186.

[5] C. Derman, On sequential decisions and Markov chains, *Management Sci.* 9 (1962) 16–24.

[6] M. Florian and P. Robillard, Hyperbolic programming with bivalent variables, Département d'Informatique, Université de Montréal, Publication #41, (August 1970).

[7] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem — Part II, *Oper. Res.* 11 (1963) 863–888.

[8] F. Glover, A new foundation for a simplified primal integer programming algorithm, *Oper. Res.* 16 (1968) 727–740.

[9] R.E. Gomory, An algorithm for integer solutions to linear programs, Princeton IBM Mathematics Research Report, (Nov. 1958), also in R.L. Graves and P. Wolfe, eds., Recent Advances in Mathematical Programming (McGraw-Hill, New York, 1963) pp. 269–302.

[10] R.E. Gomory, All integer integer programming algorithm, IBM Res. Center Report RC 189 (Jan. 1960); also in J.F. Muth and G.L. Thompson, eds., Industrial Scheduling (Prentice-Hall, Englewood Cliffs, NJ, 1963) 193–206.

[11] R.E. Gomory, An algorithm for the mixed integer problem, RAND, P-1885, (Feb. 1960).

[12] M. Grunspan and M.E. Thomas, Hyperbolic integer programming, *Naval Res. Logist. Quart.* 20 (1973) 341–356.

[13] J.R. Isbell and W.H. Marlow, Attrition games, *Naval Res. Logist. Quart.* 3 (1956) 71–93.

[14] M. Klein, Inspection–maintenance–replacement schedules under markovian deterioration, *Management Sci.* 9 (1962) 25–32.

[15] L.S. Lasdon, Optimization Theory for Large Systems; Chapters II and IV, (MacMillan London, 1970).

[16] G.T. Martin, An accelerated Euclidean algorithm for integer programming, in R.L. Graves and P. Wolfe, eds., Recent Advances in Mathematical Programming (McGraw-Hill, New York, 1963) pp. 311–317.

[17] B. Martos, Hyperbolic programming, translated by A. and V. Whinston, *Naval Res. Logist. Quart.* 11 (1964) 135–155.

[18] P. Robillard, (0, 1) hyperbolic programming problems, *Naval Res. Logist. Quart.* 18 (1971) 47–57.

[19] J. Tobin, Liquidity preference as behavior toward risk, *Rev. Economic Studies* 26 (1958) 65–86.

[20] H.M. Wagner and John S.C. Yuan, Algorithmic equivalence in linear fractional programming, *Management Sci.* 14 (1968) 301–306.

[21] H.P. Williams, Experiments in the formulation of integer programming problems, *Math. Programming Study* 2 (1974) 180–197.

[22] R.D. Young, A simplified primal (all-integer) integer programming algorithm, *Oper. Res.* 16 (1968) 750–782.

[23] W.T. Ziemba, F.J. Brooks-Hill and C. Parkan, Calculation of investment portfolios with risk free borrowing and lending, *Management Sci.* 21 (1974) 209–222.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 243-245

GRAPHS WITH CYCLES CONTAINING GIVEN PATHS

M. GRÖTSCHEL

Institut für Ökonometrie und Operations Research, D-53 Bonn, Nassestraße 2, F.R.G.

In this note we establish a sufficient condition for the following property of a graph: given any path of length s there is a cycle of length at least $m \geq s+3$ containing this path. The theorem implies the well-known theorem of Chvátal [4] on hamiltonian graphs and the theorem of Pósa [7] which gives sufficient conditions for a graph to contain cycles of a certain length. It is shown that the theorem is neither stronger nor weaker than the theorem of Bondy [2] and the still unsettled conjecture of Woodall [8].

1. Notation

The graphs $G = (V, E)$ considered are undirected, loopless, and without multiple edges. The degree $d(v)$ of a vertex $v \in V$ is the number of edges $e \in E$ containing v. A non-decreasing sequence $d_1, d_2, \ldots, d_n$ of nonnegative integers will be called a *degree sequence* if there is a graph G with n vertices $v_1, \ldots, v_n$ such that $d(v_i) = d_i$, $i = 1, \ldots, n$. A sequence $c_1, \ldots, c_n$ *majorizes* a sequence $d_1, \ldots, d_n$ if $c_i \geq d_i$, $i = 1, \ldots, n$. A sequence $P = (v_1, \ldots, v_p)$ of distinct vertices of V is called a *path* if $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \ldots, p-1$. The *length* of the path is $p - 1$. $\bar{P} = (v_p, v_{p-1}, \ldots, v_1)$ is also a path and will be called the *reverse* of P. If furthermore $\{v_1, v_p\} \in E$, P is a *cycle* of length p and will be denoted by $[v_1, \ldots, v_p]$. Sometimes we will write $[v_1, \ldots, v_p, v_1]$ instead of $[v_1, \ldots, v_p]$ for clarity. A path from v_1 to v_q, $q \leq p$, along P will be denoted by (v_1, P, v_q). If two paths $P' = (v'_1, \ldots, v'_r)$ and $P'' = (v''_1, \ldots, v''_s)$ have exactly one vertex $v'_r = v''_1$ in common then $P = (v'_1, P', v'_r, P'', v''_s)$ is a well-defined path from v'_1 to v''_s. By $N(v)$ we denote the set of neighbours of v, i.e. the set of vertices $w \in V$ such that $\{v, w\} \in E$. $|M|$ is the cardinality of a set M. $\lfloor x \rfloor$ is the greatest integer k with $k \leq x$, $\lceil x \rceil$ is the smallest integer k with $k \geq x$.

2. Properties of h-connected graphs

As a tool for further proofs we cite and prove some results concerning h-connected graphs, i.e. graphs which remain connected after the deletion of any $h-1$ vertices.

The first theorem is due to Bondy, see [1, p. 173].

Proposition 1 (Bondy). *Let G be a graph with degree sequence $d_1, \dots, d_n$ such that for some integer $h < n$ the following holds:*

$$d_k \geq k + h - 1 \quad \text{for all} \quad 1 \leq k \leq n - d_{n-h+1} - 1. \tag{1}$$

Then G is h-connected. $\square$

A well-known property of h-connected graphs is the following, cf. [1, p. 168]:

Proposition 2. *If G is h-connected then the induced subgraph obtained by removing one vertex is $(h-1)$-connected.* $\square$

The next two theorems can also be found in [1, p. 169]:

Proposition 3. *Let $G = (V, E)$ be h-connected. Let $W = \{w_1, \dots, w_h\}$ be a set of vertices, $|W| = h$. If $v \in V - W$, there exist h vertex-disjoint paths $(v, \dots, w_i)$, $i = 1, \dots, h$, joining v and W.* $\square$

Proposition 4. *Let G be a h-connected graph, $h \geq 2$. Then there is a cycle passing through an arbitrary set of two edges and $h - 2$ vertices.* $\square$

A frequently used theorem is the following, see [2, p. 192]:

Proposition 5 (Menger–Dirac). *Let $P = (a_0, a_1, \dots, a_r)$ be a path. If G is 2-connected then there exist two paths P' and P'' with the following properties:*

(a) *the endpoints of P' and P'' are a_0 and a_r,*

(b) *P' and P'' have no other points in common,*

(c) *if P' (or P'') contains vertices of P, then they appear in P' (or P'') in the same order as they do in P.* $\square$

We now give an extension of Proposition 5 which will be of interest later.

Proposition 6. *Let G be a 3-connected graph and $P = (a_0, \dots, a_r)$ be a path, let $\{a_s, a_{s+1}\}$ be an edge of this path. Then there exists a pair of paths P', P'' with the following properties:*

(a) *The endpoints of P' and P'' are a_0 and a_r,*

(b) *P' and P'' have no other points in common,*

(c) *if P' (or P'') contains vertices of P, then they appear in P' (or P'') in the same order as they do in P,*

(d) *P' contains $\{a_s, a_{s+1}\}$.*

Proof. By induction.

(1) Let $P = (a_0, a_1)$, i.e. P is an edge. Then necessarily $s = 0$. As G is 2-connected, there is another path P'' from a_0 to a_1. Take $P' = P$.

(2) $P = (a_0, a_1, a_2)$, $s = 1$. By Proposition 3 there are two vertex-disjoint paths $P_1 = (a_0, \ldots, a_1)$ and $P_2 = (a_0, \ldots, a_2)$. Define $P' = (a_0, P_1, a_1, a_2)$, $P'' = P_2$. The case $s = 0$ is similar.

(3) $P = (a_0, a_1, a_2, a_3)$, $s = 1$. By Proposition 3 there are three vertex-disjoint paths (G is 3-connected): $P_1 = (a_0, \ldots, a_1)$, $P_2 = (a_0, \ldots, a_3)$, $P_3 = (a_0, \ldots, a_2)$. Define $P' = (a_0, P_1, a_1, a_2, a_3)$ and $P'' = P_2$. All other cases are similar.

Now suppose the theorem is true for paths of length k. We prove that it is true for paths of length $k+1$.

Let $P = (a_0, a_1, \ldots, a_{k+1})$, $P_1 = (a_0, P, a_k)$.

We may assume that $s < k - 1$, otherwise we take the reverse $\bar{P}$ of P. By assumption there exist paths P_1' and P_1'' connecting a_0 and a_k having the desired properties with respect to P_1. From G we now remove the vertex a_k and add the edge $\{a_0, a_{k+1}\}$, if it does not already exist. By Proposition 2 the new graph G' is 2-connected. By Proposition 4 there is a cycle in G' containing the edges $\{a_s, a_{s+1}\}$ and $\{a_0, a_{k+1}\}$. Thus there is a path $Q = (a_0, a_1', \ldots, a_m', a_{k+1})$ in G connecting a_0 and a_{k+1} which contains the edge $\{a_s, a_{s+1}\}$ and does not contain the vertex a_k.

Let x be the vertex of path Q which is as close as possible to a_{k+1} and is contained in the union of the vertex sets P_1, P_1', and P_1''. Clearly x lies between a_{s+1} and a_{k+1} on the path Q as a_{s+1} is in Q and in P_1'. If x is in P_1' then x lies between a_{s+1} and a_k in P_1'. We now have to investigate several cases.

(i) $x = a_{k+1}$ (a) $x \in P_1'$ $P' = (a_0, P_1', x)$,
$P'' = (a_0, P_1'', a_k, a_{k+1})$.
(b) $x \in P_1''$ $P' = (a_0, P_1', a_k, a_{k+1})$,
$P'' = (a_0, P_1'', x)$.

(ii) x not in P (a) $x \in P_1'$ $P' = (a_0, P_1', x, Q, a_{k+1})$,
$P'' = (a_0, P_1'', a_k, a_{k+1})$.
(b) $x \in P_1''$ $P' = (a_0, P_1', a_k, a_{k+1})$,
$P'' = (a_0, P_1'', x, Q, a_{k+1})$.

(iii) x in P but $x \neq a_{k+1}$, say $x = a_r$, $r \geq s+1$. Let $p \leq r$ be the largest index such that a_p is contained in the union of the vertex sets of P_1' and P_1''.

(a) $a_p \in P_1'$ $P' = (a_0, P_1', a_p, P, a_r, Q, a_{k+1})$,
$P'' = (a_0, P_1'', a_k, a_{k+1})$.
(b) $a_p \in P_1''$ $P' = (a_0, P_1', a_k, a_{k+1})$,
$P'' = (a_0, P_1'', a_p, P, a_r, Q, a_{k+1})$.

These are all the cases which have to be considered and hence we are done. $\square$

Corollary 1. *Let G be $(r+2)$-connected and $P = (a_0, \ldots, a_p)$ be a path, $r \leq p$, let $Q = (a_s, \ldots, a_{s+r})$ be a path of length r contained in P. Then there exists a pair of paths P', P'' with the following properties:*

(a) *the endpoints of P' and P'' are a_0 and a_p,*

(b) *P' and P'' have no other points in common,*

(c) *If P' (or P'') contains vertices of P, then they appear in P' (P'') in the same order as they do in P.*

(d) *P' contains the path Q.*

Proof. $r = 0$: Then by definition Q is an empty path and Corollary 7 reduces to Proposition 5.

$r = 1$: This is Proposition 6.

$r > 1$: Remove the $r - 1$ vertices $a_{i+1}, a_{i+2}, \dots, a_{i+r-1}$ and add the edge $\{a_i, a_{i+r}\}$. The resulting graph G' is 3-connected by Proposition 2. The path $P_1 = (a_0, \dots, a_i, a_{i+r}, \dots, a_n)$ contains the edge $\{a_i, a_{i+r}\}$. Application of Proposition 6 gives two paths P_1' and P_1'' and P_1' contains $\{a_i, a_{i+r}\}$. The path $P' = (a_0, P_1', a_i, Q, a_{i+r}, P_1', a_n)$ is well defined in G. Define $P'' = P_1''$, then the pair P', P'' has the desired properties. □

3. The theorem and its corollaries

The following theorem establishes a sufficient condition — in terms of the degree sequence — for the following property of a graph: given any path of a specified length, there exists a cycle containing this path and having a certain minimum length. Formally the theorem is very like a theorem of Berge [1, p. 204], which is an extension of a theorem of Chvátal [4] on hamiltonian graphs. The proof of case (i) below is a slight variation of their proof which — in spirit — is due to Nash-Williams [6]. Case (ii) of the proof was motivated by Pósa's proof of his own theorem [7] which is also included in the following:

Theorem 8. *Let $d_1 \le \dots \le d_n$ be the degree sequence of a graph $G = (V, E)$. Let $n \ge 3$, $m \le n$, $0 \le r \le m - 3$, and let the following conditions be satisfied:*

$$d_k \le k + r \implies d_{n-k-r} \ge n - k \quad \text{for all} \quad 0 < k < \tfrac{1}{2}(m - r). \tag{2}$$

Furthermore, let G be $(r+2)$-connected if $\frac{1}{2}(m - r) \le n - d_{n-m+r+1} - 1$ holds and $d_k \ge k + r$ holds for all $0 < k < \frac{1}{2}(m - r)$. Then for every path Q of length r there exists a cycle in G of length at least m which contains Q.

Proof. (i) We prove: G is $(r+2)$-connected. Let $h = r + 2 \le n$, then (2) is equivalent to

$$d_k \le k + h - 2 \implies d_{n-k-h+2} \ge n - k \quad \text{for all } 0 < k < \tfrac{1}{2}(m - h + 2). \tag{2'}$$

(a) Suppose there exists a j such that $0 < j < \frac{1}{2}(m - h + 2)$ and $d_j \le j + h - 2$. Condition (2′) implies $d_{n-j-h+2} \ge n - j$. As $d_{n-m+h-1} \ge d_{n-j-h+2}$, we obtain $j \ge n - (n - j) - 1 \ge n - d_{n-m+h-1} - 1$. Thus if $d_k \le k + h - 1$, then $k \ge n - d_{n-m+h-1} - 1$. Therefore the conditions of Proposition 1 are satisfied and G is h-connected.

(b) Suppose $d_k \ge k + h - 1$ for all $0 < k < \frac{1}{2}(m - h + 2)$, then G is h-connected by Proposition 1 if $\frac{1}{2}(m - k - 2) \geq n - d_{k+1} - 1$. Otherwise k-connectedness follows from the assumption. We note for the following that $(r + 2)$-connectedness implies $d_1 \geq r + 2$.

(2) It is an easy exercise to see that a graph G' obtained from G by adding any new edge to G also satisfies (2) and the other conditions of the theorem.

(3) Suppose now that G is a graph satisfying the required conditions but which contains a path Q of length r such that Q is not contained in a cycle of length $\geq m$. By adding new edges to G we construct a "maximal" graph (also called G) which satisfies all the conditions of the theorem, contains a path Q of length r, has no cycle of length $\geq m$ containing Q, and has the property that the addition of any new edge to G creates a cycle of length $\geq m$ which contains Q. In the following we shall deal with this maximal graph G.

(4) Let $u, v \in V$ be two nonadjacent vertices of G. The addition of the edge $\{u, v\}$ will create a cycle with the desired properties. Thus there exists a path

$$P := [u_1, \ldots, u_p],\ u_1 = u,\ u_p = v,\ p \geq m$$

of length $\geq m - 1$ connecting u and v, and which contains

$$Q := (u_s, \ldots, u_{s+r}), \quad \text{where } s \in \{1, \ldots, p - r\}.$$

Let

$$S := \{i \in \{1, \ldots, p\} : \{u_1, u_{i+1}\} \in E\} \cap (\{1, \ldots, s - 1\} \cup \{s + r, \ldots, p\})$$

$$T := \{i \in \{1, \ldots, p\} : \{u_i, v\} \in E\}.$$

(a) We prove: $S \cap T = \emptyset$. Suppose $i \in S \cap T$, then $[u_1, u_{i+1}, P, u_p, u_i, P, u_1]$ is a cycle with the desired properties. Contradiction!

(b) $|S| + |T| \leq |P| - 1$ because $p \notin S \cup T$.

(5) The degree sequence of G necessarily has exactly one of the following properties:

Case (i) there is a k_0, $0 < k_0 < \frac{1}{2}(m - r)$, such that $d_{k_0} \leq k_0 + r$.

Case (ii) $d_k > k + r$ for all $0 < k < \frac{1}{2}(m - r)$.

These cases will be handled separately.

Case (i)

(6) As $d_1 \geq r + 2$ and as the degree sequence $d_1, \ldots, d_n$ is increasing there is a $j \leq k_0$ such that $d_j = j + r$. (2) implies $d_{n-j-r} \geq n - j$, i.e. there are $j + r + 1$ vertices of V having degree at least $n - j$. The vertex having degree $j + r$ cannot be adjacent to all of these. Thus there exist two nonadjacent vertices $a, b \in V$ such that $d(a) + d(b) \geq n + r$.

(7) Among all nonadjacent vertices of G choose u, v such that $d(u) + d(v)$ is as large as possible. Define P, S, T, Q as in (4). We calculate $d(u) + d(v)$. Obviously

$$d(v) = |T| + \alpha \quad \text{where } \alpha \leq |V - P|$$

and

$$d(u) \leq |S| + r + \beta \quad \text{where } \beta \leq |V - P|.$$

Suppose there is a $w \in V - P$ which is adjacent to both u and v. Then $[u_1, u_2, \ldots, u_p, w]$ would be a desired cycle. Therefore $\alpha + \beta \le |V - P|$, which leads, using (4) (a) and (b), to

$$
\begin{aligned}
d(u) + d(v) &\le |T| + \alpha + |S| - r + \beta \\
&\le |P| - 1 + \alpha + \beta + r \\
&\le |P| + |V - P| + r - 1 \\
&\le n + r - 1.
\end{aligned}
$$

By (6) $d(u) + d(v)$ cannot be maximal. Contradiction!

Case (ii).

(8) Among all longest paths in G containing Q choose a path such that the sum of the degrees of the endpoints is as large as possible. As Q is maximal, the length of this path is at least $m - 1$, and the endpoints are not joined by an edge. Let this path be $P = (u_1, \ldots, u_p)$ and Q, T, S be defined as in (4).

(9) We prove: $d(u_1) \ge \frac{1}{2}(m + r)$, $d(u_p) \ge \frac{1}{2}(m - r)$. Suppose $d(u_1) < \frac{1}{2}(m + r)$. All neighbours of u_1 and u_p are contained in P, otherwise P would not have maximal length. As $d_1 \ge r + 2$, we have $d(u_1) \ge r + 2$ and therefore $|S| \ge d(u_1) - r \ge 1$. All vertices u_i, $i \in S$, have degree at most $d(u_1)$, otherwise $(u_i, u_{i-1}, \ldots, u_1, u_{i+1}, u_{i+2}, \ldots, u_p)$ would be a path of the same length as P and $d(u_i) + d(u_p) > d(u_1) + d(u_p)$, contradicting the maximality assumption on the endpoints of P. Let $j_0 := d(u_1)$, then there are $|S| \ge j_0 - r$ vertices of degree at most j_0. As we are in case (ii), $d_k > k + r$ holds for all $0 < k < \frac{1}{2}(m - r)$, which is equivalent to $d_{j-r} > j$ for all $r < j < \frac{1}{2}(m + r)$. Therefore $j_0 \ge \frac{1}{2}(m + r)$. By similar arguments $d(u_p) \ge \frac{1}{2}(m + r)$.

(10) From (9) it follows that

$$|S| + r + |T| \ge d(u_1) + d(u_p) \ge m + r.$$

Thus $|S| + |T| \ge m$, and from (4) (b) we have $|P| \ge m + 1$. Therefore if $m = n$ we have $n \ge |P| \ge n$ which is a contradiction, and in this case we are done.

(11) Let $N := N(u_1) \cup N(u_p) \cup \{u_1, \ldots, u_{r+1}\} \cup \{u_1, u_p\}$. We prove: $|N| \ge m$. As $r \le m - 3$, $|\{u_2, \ldots, u_{r+1}\} \cap \{u_1, u_p\}| = 1$.

(a) Suppose $\max\{i \in S\} \le \min\{j \in T'\}$, where $T' := T - \{s, \ldots, s + r\}$. This means that the index of a neighbour of u_1 which is not among $u_2, \ldots, u_{r+2}$ is less than or equal to the smallest of the indices of the neighbours of u_p not among $u_2, \ldots, u_{r+2}$. Thus $|(N(u_1) \cap N(u_p)) - \{u_2, \ldots, u_{r+2}\}| \le 1$. Obviously

$$
\begin{aligned}
|N| \ge {} & |N(u_1) - \{u_2, \ldots, u_{r+2}\}| + |N(u_p) - \{u_2, \ldots, u_{r+2}\}| \\
& + |\{u_2, \ldots, u_{r+2}\}| + |\{u_1, u_p\}| - |(N(u_1) \cap N(u_p)) - \{u_2, \ldots, u_{r+2}\}| \\
& - |\{u_2, \ldots, u_{r+2}\} \cap \{u_1, u_p\}| \\
\ge {} & |S| - 1 + |T| + (r + 1) + 2 - 1 - 1 \\
\ge {} & |S| + |T| \ge m.
\end{aligned}
$$

(b) Suppose $\max\{i \in S\} > \min\{j \in T'\}$. Let

$$d := \min\{(i+1) - j : i \in S,\ j \in T' \text{ such that } i > j\},$$

then we have $d > 0$. Now let $i_0 + 1 - j_0 = d$.

(b_1) $i_0 + 1 \le s$. By definition $j_0 < s$ and no vertex of the path P between u_{j_0} and u_{i_0+1} is linked to u_1 or u_p by an edge. Thus

$$[u_1, u_{i_0+1}, u_{i_0+2}, \ldots, u_p, u_{j_0}, u_{j_0-1}, \ldots, u_1]$$

is a cycle containing the path Q, all vertices u_i, $i \in S$, with the possible exception of $i = i_0$, and all vertices u_j, $j \in T'$. It also contains u_1 and u_p. Thus the length of this cycle is at least:

$$(r-2) + |S| - 1 + |T'| \ge |S| + |T'| \ge m$$

which is impossible by assumption.

(b_2) $r - s < j_0$. Define the same cycle as in (b_1) and by the same arguments we obtain a contradiction.

(b_3) $j_0 < s$, $i_0 > r - s$. Define

$$j_1 := \min\{j \in T'\} < j_0, \qquad i_2 := \max\{i + 1 : i \in S\} > r - s + 1.$$

The conditions of case (b_3) imply the following:

$$u_1 \not\sim u_n, \ u_p \not\sim u_{r-s},$$

none of the vertices u_i, $j_1 < i \le s$, can be linked to u_1 by an edge, none of the vertices u_i, $i_2 < i \le p$, is a neighbour of u_1, thus

$$N(u_1) \subset \{u_2, \ldots, u_{j_1}\} \cup \{u_{s+1}, \ldots, u_{i_2}\},$$

none of the vertices u_i, $1 \le i < j_1$, is a neighbour of u_p, none of the vertices u_i, $r - s < i < i_2$, is a neighbour of u_p, thus

$$N(u_p) \subset \{u_{j_1}, \ldots, u_{r-s}\} \cup \{u_{i_2}, \ldots, u_{p-1}\}.$$

Furthermore

$$|N(u_1) - \{u_{s+1}, \ldots, u_{i_2}\}| = |S|,$$

$$|N(u_p) - \{u_{i_2}, \ldots, u_{p-1}\}| = |T'|.$$

The only vertices which might be neighbours of both u_1 and u_p are u_{j_1}, u_{i_2} and $u_{s+1}, \ldots, u_{r-s}$. This implies

$$|(N(u_1) \cap N(u_p)) - \{u_{s+1}, \ldots, u_{r-s}\}| \le 2.$$

Therefore

$$|N| \ge |N(u_1) - \{u_s, \ldots, u_{r-s}\}| - |N(u_p) - \{u_s, \ldots, u_{r-s}\}| - (r+1) + 2 - 2 - 1$$

$$\ge |S| + |T'| + r$$

$$\ge |S| + |T'| \ge m.$$

These are all the cases which can occur, therefore $|N| \geq m$ is proved.

(12) Among all pairs of paths satisfying Corollary 7 with respect to P and Q choose a pair P^1, P^2 such that the cycle $K = [u_1, P^1, u_p, \bar{P}^2, u_1]$ contains as many vertices of P as possible.

(13) To show that K has length $\geq m$, we will prove: K contains all vertices of N. Suppose there is a vertex of N which is not contained in K. Trivially the vertex is either in $N(u_1) - \{u_2, \dots, u_{r-1}\}$ or in $N(u_p) - \{u_{s+1}, \dots, u_{p-1}\}$. Without loss of generality we assume that the vertex $u_k \in N(u_1) - \{u_2, \dots, u_{r-1}\}$ is not contained in K. Let

$$i_0 = \max\{i \mid u_i \in N \cap K,\ i < k\} \qquad j_0 = \min\{i \mid u_i \in N \cap K,\ i > k\}.$$

(a) Suppose $u_{i_0}, u_{j_0} \in P^1$, then

$$P_1^1 = (u_1, P^1, u_{i_0}, P, u_{j_0}, P^1, u_p),$$

$$P_1^2 = P^2,$$

is a pair of paths satisfying Corollary 7, and $K_1 = [u_1, P_1^1, u_p, \bar{P}_1^2, u_1]$ contains more vertices of P then K does. Contradiction! If $u_{i_0}, u_{j_0} \in P^2$ the contradiction follows similarly.

(b) Suppose $u_{i_0} \in P^1$, $u_{j_0} \in P^2$. Let

$$P_2^1 = (u_1, P, u_{i_0}, P^1, u_p),$$

$$P_2^2 = (u_1, u_k, P, u_{j_0}, P^2, u_p).$$

If $j_0 \leq s$ then Q is contained in (u_{j_0}, P^2, u_p), otherwise Q is contained in (u_1, P, u_{i_0}). Therefore P_2^1 and P_2^2 satisfy the conditions of Corollary 7, and K_2 contains more vertices of P then K does. Contradiction!

(c) Suppose $u_{i_0} \in P^2$, $u_{j_0} \in P^1$.

(c_1) $i_0 \leq s$: this implies $j_0 \leq s$.

Take

$$P_3^1 = (u_1, u_k, P, u_{j_0}, P^1, u_p)$$

$$P_3^2 = (u_1, P, u_{i_0}, P^2, u_p).$$

(c_2) $i_0 \geq r - s$:

Let

$$P_4^1 = (u_1, P, u_{i_0}, P^2, u_p),$$

$$P_4^2 = (u_1, u_k, P, u_{j_0}, P^1, u_p).$$

These pairs of paths satisfy Corollary 7. The contradiction follows as above.

Thus in Case (ii) we have constructed a cycle K of length $\geq m$ containing the path Q, which contradicts the assumption that G does not contain such a cycle, and we are done. $\square$

Theorem 8 has some immediate Corollaries and also includes some of the classical theorems on graphs containing cycles of a certain minimum length.

Corollary 9. *Let $d_1, \ldots, d_n$ be the degree sequence of a graph $G = (V, E)$. Let $n \geq 3$, $q \geq 2$ and let the following condition be satisfied:*

$$d_k \leq k \leq q-1 \Longrightarrow d_{n-k} \geq n-k. \tag{3}$$

Furthermore, let G be 2-connected if $q-1 \leq n-d_{n-1}-1$ holds and $d_k > k$ holds for all $1 \leq k \leq q-1$. Then G contains a cycle of length at least $\min\{n, 2q\}$.

Proof. Take $r = 0$ in Theorem 8. □

One of the well-known theorems implied by Theorem 8 is the following due to Pósa [7], which generalizes results of Dirac [5].

Corollary 10 (Pósa [7]). *Let $d_1, \ldots, d_n$ be the degree sequence of a 2-connected graph G. Let $q \geq 2$, $n \geq 2q$. If*

$$d_k > k \quad \text{for all } k = 1, \ldots, q-1, \tag{4}$$

then G contains a cycle of length at least $2q$.

Proof. Immediate from Corollary 9. □

For bipartite graphs a simple trick yields:

Corollary 11. *Let $G = (V, W, E)$ be a bipartite graph with degree sequences $d(v_1) \leq \cdots \leq d(v_n)$ and $d(w_1) \leq \cdots \leq d(w_m)$, $n \leq m$. If*

$$d(w_k) \leq k \leq n-1 \Longrightarrow d(v_{n-k}) \geq m-k+1, \tag{5}$$

then G contains a cycle of length $2n$.

Proof. Construct $G^* = (V \cup W, E^*)$ by adding all edges to E which have both endpoints in V. Clearly G^* contains a cycle of length $2n$ if and only if G does. If G satisfies (5) then G^* satisfies (3). As (5) implies that $d(w_1) \geq 2$ and V defines a clique in G^*, G^* is 2-connected. □

Standard theorems giving sufficient conditions for a graph to be hamiltonian can also be derived from Theorem 8.

Corollary 12 (Berge, [1, p. 204]). *Let $G = (V, E)$ be a graph with degree sequence $d_1, \ldots, d_n$. Let r be an integer, $0 \leq r \leq n-3$. If for every k with $r < k < \frac{1}{2}(n+r)$ the following condition holds:*

$$d_{k-r} \leq k \Longrightarrow d_{n-k} \geq n-k+r, \tag{6}$$

then for each subset Q of edges, $|Q| = r$, that forms a path there is a hamiltonian cycle in G that contains Q.

Proof. Clearly (6) is equivalent to (2) if $m = n$. We have to prove that (6) implies $(r+2)$-connectedness.

If there is a k with $r < k < \frac{1}{2}(n+r)$ such that $d_{k-r} \leq k$, then by the arguments of the proof of Theorem 8, Section (I) (a) $(r+2)$-connectedness is assured.

If $d_{k-r} > k$ for all $r < k < \frac{1}{2}(n+r)$, we have $d_q \geq q + r$, where $q := \left\lceil \frac{n-r}{2} \right\rceil$. Furthermore $2q \geq n - r$ and $q \leq n - r - 1$ (as $r \leq n-3$), thus $q + r \leq d_q \leq d_{n-q}$. This implies

$$q = 2q - q \geq n - (r+q) > n - (q+r) - 1 \geq n - d_{n-q} - 1.$$

Thus condition (1) of Proposition 1 is satisfied and G is $(r+2)$-connected. □

Actually Berge proved a stronger theorem saying that Q only has to be a set of edges of cardinality r such that the connected components of Q are paths.

Corollary 13 (Chvátal [4]). *If the degree sequence $d_1, \ldots, d_n$ of a graph G, $n \geq 3$, satisfies*

$$d_k \leq k < \tfrac{1}{2}n \implies d_{n-k} \geq n - k, \tag{7}$$

then G contains a hamiltonian cycle.

Proof. Take $r = 0$ in Corollary 12. □

Furthermore, Chvátal showed that this theorem is best possible in the sense that if there is a degree sequence of a graph not satisfying (7) then there exists a non-hamiltonian graph having a degree sequence which majorizes the given one. This proves that Theorem 8 is also best possible in this special case. Moreover Chvátal (see [4]) showed that most of the classical results on hamiltonian graphs are contained in his theorem, and therefore are also implied by Theorem 8.

A trivial consequence of Corollary 13 which however is not too "workable" is

Corollary 14. *Let G' be an induced subgraph of a graph G having $m \leq n$ vertices. If the degree sequence $d'_1, \ldots, d'_m$ of G' satisfies (7) then G contains a cycle of length m.* □

4. Some examples

(a) We first show that the number m implied by Theorem 8 giving the minimum length of a cycle containing a given path cannot be increased, i.e. we give an example of a graph G with a path Q of length r such that the longest cycle containing Q has length m.

Consider a graph with two disjoint vertex sets A and B. A is a clique of q

vertices, and B consists of p isolated vertices. Each vertex of A is linked to each vertex of B by an edge. Suppose that $1 \leq q - r$ and $p \geq q - r + 1$. The degree sequence of G is

$$\underbrace{q, q, \ldots, q}_{p \text{ times}}, \underbrace{n-1, \ldots, n-1}_{q \text{ times}}.$$

Hence we have

$$d_i \geq i + r \quad \text{for } i < q - r,$$

$$d_{q-r} = (q - r) + r = q,$$

$$d_{n-(q-r)-r} = d_{n-q} = q < q + 1 = (2q - r) + 1 - (q - r) \leq n - (q - r).$$

By Theorem 8 for each path Q of length r there is a cycle of length $2q - r$ containing Q.

If we choose a path Q of length r such that all vertices of Q are contained in A it is obvious that no longer cycle containing Q exists.

(b) We give an example showing that the assumption of $(r+2)$-connectedness in Theorem 8 under the specified conditions is necessary.

Consider the graph G consisting of three vertex sets A, B, C. A and B have k vertices and are complete, C has $r + 1$ vertices and is complete. Each vertex of C is joined to each vertex of $A \cup B$ by an edge. Hence G is $(r+1)$-connected but not $(r+2)$-connected. Take a path Q of length r in C. Clearly the maximal length of a cycle containing Q is $k - r + 1$. The degree sequence of this graph is

$$\underbrace{k + r, \ldots, k + r}_{2k \text{ times}}, \underbrace{n - 1, \ldots, n - 1}_{r+1 \text{ times}}.$$

We have $d_i \geq i + r$ for $0 < i \leq k - 1$, therefore Theorem 8 would imply the existence of a cycle of length at least $2k + r$ containing Q.

(c) We give an example showing that Corollary 14 is not stronger than Corollary 9.

Consider a graph consisting of two disjoint cliques A, B, each having m vertices. Link A and B by two disjoint edges. Obviously this graph is hamiltonian. The degree sequence is

$$\underbrace{m - 1, \ldots\ldots, m - 1}_{2m-4 \text{ times}}, m, m, m, m.$$

Corollary 9 implies that there exists a cycle of length $\geq 2m - 2$, but Corollary 14 does not imply a cycle of length $\geq 2m - 2$.

(c$_1$) Delete 2 vertices of A, both must necessarily be distinct from the two vertices linking A to B. The degree sequence is

$$\underbrace{m-3,\dots,m-3}_{m-4 \text{ times}}, m-2, m-2, \underbrace{m-1,\dots,m-1}_{m-4 \text{ times}}, m, m$$

which does not satisfy (7).

(c$_3$) Delete one vertex of A and one of B, again both must be distinct from the vertices linking A to B. The degree sequence is

$$\underbrace{m-2,\dots\dots,m-2}_{2(m-3) \text{ times}}, m-1, m-1, m-1, m-1$$

which also does not satisfy (7).

It is clear that Corollary 9 does not imply Corollary 14.

(d) Bondy proved (see [3]) the following

Theorem (Bondy). *Let G be a 2-connected graph with degree sequence $d_1,\dots,d_n$. If*

$$d_j \le j,\ d_k \le k\ (j \ne k) \Longrightarrow d_j + d_k \ge c, \tag{8}$$

then G has a cycle of length at least $\min(c, n)$. □

Chvátal showed that in the case $c = n$ his theorem (Corollary 13) implies Bondy's theorem, thus in the hamiltonian case Corollary 9 is stronger than the theorem of Bondy. In general this is obviously not true, nor is the converse as the following example shows: The graph has three vertex sets A, B, C. $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$, $|C| = m$. The edges are the following: $\{a_1, b_1\}$, $\{a_1, b_2\}$, $\{a_2, b_1\}$, $\{a_2, b_2\}$, $\{a_3, b_3\}$, $\{a_3, b_4\}$, $\{a_2, b_4\}$, and all edges having both endpoints in $B \cup C$. The degree sequence is

$$2, 2, 3, \underbrace{n-4,\dots,n-4}_{m \text{ times}}, n-3, n-2, n-2, n-2.$$

$d_2 \le 2$ and $d_3 \le 3$. By Pósa's theorem there is a cycle of length ≥ 4, by Bondy's theorem there exists a cycle of length ≥ 5. As $d_{n-2} \ge n-2$ and $d_{n-3} \ge n-3$ and $d_i \ge i$, $4 \le i \le \frac{1}{2}n$, G is hamiltonian by Corollary 9.

(e) In [8] Woodall stated the following (to my knowledge unsettled)

Conjecture. *Let $d_1,\dots,d_n$ be the degree sequence of a 2-connected graph G, $m \le n-3$, and let the following condition be satisfied:*

$$\begin{cases} d_{k+m} > k & \text{for } 1 \le k < \frac{1}{2}(n-m-1), \\ d_{k+m+1} > k & \text{if } k = \frac{1}{2}(n-m-1). \end{cases} \tag{9}$$

Then G contains a cycle of length at least $n-m$. □

Obviously Corollary 9 does not imply Woodall's Conjecture, but surprisingly not does the Conjecture imply Corollary 9, although in most cases Woodall's Conjecture—if true—would be "better" than Corollary 9.

We give an example: Let n and m be both odd (or even), $j = \frac{1}{2}(n - m - 2)$ and $j^2 \geq \frac{1}{2}(n - m)$ (which is a solvable condition).

Consider the following graph consisting of three vertex sets A, B, $\{u\}$. B has $j + 1$ elements and is complete, u is linked to all elements of B by an edge. A consists of $j + m$ isolated vertices, each element of A is linked to exactly j vertices of B such that each element of B is linked to at least $m + 1$ vertices of A. This is possible as $(j - m)j = jm - j^2 \geq jm - \frac{1}{2}(n + m) = jm + j - m + 1 = (m - 1)(j + 1)$. The degree sequence of this graph is

$$\underbrace{j, \dots, j}_{j-m \text{ times}}, j+1, \underbrace{m_1, \dots, m_{j+1}}_{j+1 \text{ times}}$$

where $m_i \geq n - j$ for $i = 1, \dots, j + 1$. We have

$$d_{k+m} > k \quad \text{for } 1 \leq k \leq j - 1,$$
$$d_{j+m} = j \quad \text{and} \quad j < \tfrac{1}{2}(n - m - 1)$$

Thus Woodall's Conjecture does not imply a cycle of length $\geq n - m$. On the other hand

$$d_k > k \quad \text{for } 1 \leq k \leq j - 1,$$

$$d_j = j \quad \text{and} \quad d_{n-j} = m_1 \geq n - j.$$

Hence by Corollary 9 there exists a cycle of length $\geq 2(j + 1) = n - m$.

References

[1] C. Berge, *Graphs and Hypergraphs* (North Holland, Amsterdam, 1973).
[2] C. Berge, *Théorie des Graphes et ses Applications* (Dunod, Paris, 1958).
[3] J.A. Bondy, Large cycles in graphs, *Discrete Math.* 1 (1971) 121–132.
[4] V. Chvátal, On Hamilton's ideals, *J. Combinatorial Theory* 12 B (1972) 163–168.
[5] G.A. Dirac, Some theorems on abstract graphs, *Proc. London Math. Soc.* 3 (2) (1952) 69–81.
[6] C.St.J.A. Nash-Williams, On hamiltonian circuits in finite graphs, *Proc. Am. Math. Soc.* 17 (1966) 466–467.
[7] L. Pósa, On the circuits of finite graphs, *Publ. Math. Inst. Hung. Acad. Sc.* 8 (1963) 355–361.
[8] D.R. Woodall, Sufficient conditions for circuits in graphs, *Proc. London Math. Soc.* (3) 24 (1972) 739–755.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 247-271

ALGORITHMS FOR EXPLOITING THE STRUCTURE OF THE SIMPLE PLANT LOCATION PROBLEM

Monique GUIGNARD

Department of Statistics, Wharton School, University of Pennsylvania, Philadelphia, PA 19174, U.S.A.

Kurt SPIELBERG

Scientific Marketing, IBM, White Plains, NY 10604, USA

The paper is concerned with a number of approaches to the important simple plant location problem. In addition to describing several decomposition approaches, the paper focuses on modified simplex methods which exploit triangular bases.

1. Introduction

Simple (Uncapacitated) Plant Location Problems (we shall abbreviate Simple Plant Location by SPL, and SPL problem by SPLP) are of great significance both practically and theoretically. There exist telecommunication network problems which could use algorithms handling problems with thousands of "plants" and "destinations". These can only be tackled by heuristics at present.

The SPLP is one of the simplest mixed integer problems which exhibit all the typical combinatorial difficulties of mixed (0, 1) programming and at the same time have a structure that invites the application of various specialized techniques.

1.1. *Brief survey*

The referee's comments about the literature on SPL and related problems, for which we express our appreciation, indicate that a brief survey of some of the literature is necessary, incomplete as it must be for such a big subject.

Exact formulations appear to go back to Balinski [4]. A first enumerative algorithm of the branch-bound type, based on the "*aggregated*" constraints $\sum_j x(i,j) \le m(i) \cdot y(i)$, was developed by Efroymson, Ray in [13]. It was later refined by a number of authors.

But the current state of the art must almost certainly rest squarely on the resolution of the SPLP with "*disaggregated*" constraints $x(i,j) \le y(i)$, because the "relaxed" problem with $0 \le y(i) \le 1$ is very strong for the disaggregated and very weak for the aggregated form. This notion appears to have been observed and exploited independently by three groups of researchers.

Bilde and Krarup, in a paper published in Danish (in 1967), and therefore unfortunately largely unread (available now in [7]), devised excellent heuristic techniques for producing strong lower bounds on the objective function of the strong relaxed problem, exploited with good effect in a branch and bound algorithm.

A class of enumerative algorithms by Spielberg [24, 25], based on widely distributed IBM reports of 1967 and 1968, exploited the disaggregated form in terms of dual variable analysis leading to strong Benders inequalities and "*gain functions*". The work was extended to more general problems in Guignard, Spielberg [20]. A recent paper by Cornuejols, Fischer, Nemhauser [10] analyzes nicely a "greedy algorithm" which is based on one of the algorithms of [25] and has the additional merit of establishing clearly (by way of Lagrangean Techniques, due to Held and Karp and extended and summarized by Geoffrion [15]) that the disaggregated form of the constraints is indeed fully exploited in this fashion.

The third important approach (expressed in terms of the capacitated problem) is due to Davis, Ray [12], who solved the linear program by decomposition in 1967. This work established the practicability and desirability of solving the disaggregated LP directly.

What lends special interest to the above is that there has been steadily increasing recognition of the importance of disaggregation for large scale problems in the entire class of location and distribution problems, an area whose practical importance can hardly be overstated.

Without being in any sense complete, we can cite work on the M Median Plant Location Problem by Garfinkel, Neebe, Rao [18], a successful application of Benders' algorithm to a large distribution problem by Geoffrion, Graves [16], and a general account of formulation techniques by Williams [26].

Finally we have recently seen the resolution of quite large distribution problems, with several thousand integer variables, by the general purpose code MPSX-MIP of IBM, after suitable introduction of disaggregated constraints (e.g., E.L. Johnson, private communication).

1.2 *Approach of current paper*

The following paper focuses first on decomposition and then on new possibilities for exploitation of the fully disaggregated linear program. In the latter area one might also consult the work of Marsten [22] and Graves, McBride [19] on specialized Simplex Methods.

Recent papers of Schrage [23] on implicit representation of generalized variable upper bounds, and Glover [17] on compact LP bases provide general techniques for problems which we called "weakly linked" in [3] and [20], a class of problems which encompasses location and more general fixed charge problems.

Finally, it may be of interest that there is a link to the Russian literature via the two references Frieze [14] and Babayev [1].

The first paper demonstrates a property for the gain functions of [24, 25], and the second relates this property to a "method of successive calculation" of Cherenin [9].

It is always enticing to start by decomposition techniques in order to get good bounds on the objective function. Equally interesting is the construction of specialized simplex algorithms which attempt to adhere to the great abundance of all integer vertices as much as is at all possible. We have been able to solve a (20, 35) SPLP by a linear programming triangularization method, carefully bypassing all fractional vertices which would naturally lie in the path of an unmodified primal algorithm.

Such attempts have been given new impetus by the results of Balas and Padberg, given in [2, 3], to the effect that there is always a path of integer vertices leading to the integer optimum of a SPLP. This is a nice result, but an algorithm such as suggested in [3] runs into formidable difficulties which appear to be very much of an enumerative nature.

As opposed to the "usual" set-packing problem treated in [2, 3], the SPLP is unusual in the sense that as linear programming (LP) problem it is enormously large for problems which must be considered small in practice.

To tackle the SPLP successfully, then, one must have highly specialized tools for treating everything within the LP problem implicitly. In Section 3 we discuss a certain special basis representation, which we believe must play a role (possibly in a yet somewhat more modified form) in any efficient direct linear programming treatment of the SPLP.

Actually, we believe that Section 3 is important in several respects. The possibility of constructing triangular bases which lead to easily obtainable updated tableaux can be exploited for writing *computationally efficient* codes for problem sizes which would otherwise be intractable. What may be just as important, the latitude in constructing such bases can apparently be exploited to render them "*good*", in the sense of minimizing the number of negative reduced costs (related to gain functions which have been found to be important elsewhere).

Finally, these triangularization procedures are such that they can be applied to *any integer feasible solution*, no matter how it was found. This opens the way to a class of algorithms, dependent on the actual triangularization process adopted, consisting of steps such as:

(1) Heuristics, enumeration, etc., to give a feasible integer solution.

(2) Construction of a "good" triangular basis, and therefore a simply structured (implicit) updated tableau.

(3) Exploration of neighbor vertices. Pivot or block pivot to neighbor vertex.

(4) Return to (1).

I.e., depending on the actual basis choices, one has a class of true hybrid algorithms, which are LP intermittently, but then also permit *jumps* from one lattice point to completely different lattice points without loss in efficiency of LP computation (e.g., given a solution point (x, y) arrived at by a LP step, it is possible that x can be improved for given y by inspection, or the jump might correspond to one of the simple heuristics which are easily available for SPLP). Notice that tableaux are never updated, since bases and inverses are easily constructed from the solution.

2. Decomposition and partitioning methods

There are many possibilities of decomposition and partitioning. On balance they are by now quite well known. We believe that the "*reverse partitioning*" of 2.2.3 is new, somewhat unusual, and therefore interesting.

What is most important, however, is the potential utilization of the special problem characteristics. It is clear that the difference between success and failure lies here, and we have tried to present ideas which might form a start for a real algorithm (stand-alone or auxiliary algorithm within enumeration).

$$\begin{aligned} &\min z = \sum f(i)\cdot y(i) + \sum c(i,j)\cdot x(i,j),\\ &\sum x(i,j) = 1, \quad \text{all } j,\\ &x(i,j) \le y(i), \quad \text{all } i,j,\\ &y(i) = 0 \text{ or } 1, \quad x(i,j) \ge 0. \end{aligned} \tag{2.1}$$

The indices i and j range from 1 to m, and 1 to n, respectively. We admit only $f(i) \ge 0$ and $c(i,j) \ge 0$. Whether we consider (2.1) or its relaxed LP form (all $y(i)$ between 0 and 1) will usually be clear from the context.

2.1. *Dantzig and Wolfe decomposition* [11]

Consider the relaxed problem.

$$\begin{aligned} &\min fy + cx \quad (= gt)\\ &\text{s.t.} \sum x_{ij} = 1 \quad (At = 1),\\ &\left.\begin{aligned} &-y_i + x_{ij} \le 0\\ &1 \ge y_i, \quad y_i \ge 0 \end{aligned}\right\} (Bt \le 0), \end{aligned}$$

where $g = (f, c)$, $t = \binom{y}{x}$. Let $t^1, t^2, \ldots, t^k, \ldots$ be the extreme points of $\mathcal{B} = \{t \in \mathbb{R}^{m+mn} \mid Bt \le 0\}$ (a compact set), and let K be the set of their indices. Then, for all $t \in \mathcal{B}$, there exists $\lambda = (\lambda_1, \ldots, \lambda_k, \ldots)$ such that

$$\lambda \geq 0,$$

$$\sum_{k \in K} \lambda_k = 1,$$

$$t = \sum_{k \in K} \lambda_k t^k.$$

Then the decomposition algorithm will consider the following two problems:
(A–P)

$$\min_{\lambda} \sum_{k \in K'} \lambda_k (gt^k) - \sum_{k \in K'} \lambda_k z^k$$

$$\text{s.t.} \sum_{k \in K'} \lambda_k (A_j t^k) = 1, \quad j = 1, \ldots, m,$$

$$\sum_{k \in K'} \lambda_k = 1, \quad \lambda_k \geq 0, \forall k \in K',$$

where K' is the index set of currently known vertices of $\mathcal{T}$, with the dual
(A–D)

$$\bar{z} = \max_{u, v} \sum_{j=1}^{m} u^j - v$$

$$-gt^k + \sum_{j=1}^{m} u^j (A_j t^k) + v \leq 0$$

and the problem
(B–P)

$$\hat{d} = \min_{t} \left(gt - \sum_{j=1}^{m} u^j A_j t - v - d \right)$$

$$\text{s.t. } t \in \mathcal{T}, \text{ i.e. } \begin{cases} 0 \leq x_{ij} \leq y_i, & \text{all } i, j, \\ 0 \leq y_i, & \text{all } i. \end{cases}$$

In fact, (A–P) and (B–P) can be rewritten
(A–P)

$$z = \min_{\lambda} \sum_{k \in K} \lambda_k \left(\sum_i f_i y_i^k + \sum_{ij} c_{ij} x_{ij}^k \right) - \sum \lambda_k z^k$$

$$\text{s.t.} \sum_{k \in K'} \lambda_k \left(\sum_i x_{ij}^k \right) = 1, \quad j = 1, \ldots, n$$

$$\sum_{k \in K'} \lambda_k = 1, \quad \lambda_k \geq 0 \text{ all } k \in K'.$$

(B–P)

$$\hat{d} = \min_{x, y} \left(\sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij} - \sum_j u^j \left(\sum_i x_{ij} \right) - v - d \right)$$

$$= \sum_i f_i y_i - \sum_i \sum_j (c_{ij} - u_j) x_{ij} - u,$$

$$0 \leq x_{ij} \leq y_i,$$

$$0 \leq y_i \leq 1.$$

The continuous objective function of SPLP is in the bracket:

$$\underline{z} + d \leq \hat{z} \leq \bar{z}$$

It is well known [8] that the algorithm converges even when $(B-P)$ is not optimized, but suboptimized, i.e. as long as the solution $r = (y, x)$ is chosen so as to render d negative and to be an extreme point of $\mathcal{S}$. Also, as long as there are feasible integer solutions to SPLP whose objective function values are between $\underline{z}$ and the current best value $z^* = \min z$, every such solution is not yet included in the set of generators of $\mathcal{S}$ and would yield an improvement over the current $\sum \lambda_k r^k$, i.e. yield a negative value for d. If the optimal solution is not integral, the integer optimum is among those feasible solutions that render the last d negative.

Also, if an improving $(d < 0)$ feasible solution to SPLP is the only generator added at that iteration, it will be the optimum of the next $(\Lambda-P)$ problem. Yet it might be better to add both the optimum and a feasible solution of SPLP simultaneously.

Remark 1. Every time a $(\Lambda-P)$ problem is solved, its solution yields a new feasible solution to the original LP. It is of the form (y, x). Keeping in mind the original problem, one may be able to find a better solution by taking y' defined by

$$y'(i) = \max_j x(i, j),$$

the new cost $f.y' + c.x$ being no larger than $f.y + c.x$. This is important, since $f.y + c.x$ (or $f.y' + c.x$) is an upper bound for the optimal value of the original problem. If (y', x) is an extreme point of $\mathcal{S}$, one can add it to the current set of generators.

Remark 2. The constraints of a $(B-P)$ problem are such that the problem is separable, as the constraints which enforce the presence of exactly one $x(i, j) = 1$ per column have disappeared from its formulation. Each $(B-P)$ yields m subproblems of the form

$$\min f_i y_i - \sum_j (c_{ij} - u_j) x_{ij} - u/m,$$

$$\text{s.t. } 0 \leq x_{ij} \leq y_i \leq 1,$$

whose solution is obvious. If $c(i, j) - u(j) \geq 0$, set $x(i, j)$ to 0. Then, if there are some $c(i, j) - u(j) < 0$, set $x(i, j)$ equal to, say, α. Thus, $y(i)$ must be at least equal to α. The objective function is then equal to $(f(i) + \sum_{j \in J^-} (c(i, j) - u(j))) \cdot \alpha$. If the coefficient of α is negative, set $\alpha = 1$, otherwise set $\alpha = 0$. One can follow this by an attempt to find a suboptimal, feasible solution to SPLP such that $d < 0$.

Remark 3. An initial set of extreme points of $\mathcal{R}$ should be carefully chosen to allow generation of meaningful points from the outset. For instance, one might choose

$$r^0 = 0 \text{ all } i, j;\ r^1 = 1 \text{ all } i, j;\ r^2, \ldots, r^{k+1}, \ldots, r^{n+1}$$

such that there is one 1 in column k of x, corresponding to the smallest $c(i, j)$, the corresponding $y(i)$ being set equal to 1, all others to 0.

Example. The following data

$$f = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad c = \begin{bmatrix} 1 & 1 & 10 \\ 1 & 10 & 1 \\ 10 & 1 & 1 \end{bmatrix}$$

yield a continuous optimal solution

$$y = \begin{bmatrix} \frac12 \\ \frac12 \\ \frac12 \end{bmatrix}, \qquad x = \begin{bmatrix} \frac12 & \frac12 & 0 \\ \frac12 & 0 & \frac12 \\ 0 & \frac12 & \frac12 \end{bmatrix}, \quad z = 4.5$$

and several integer optimal solutions among them,

$$y = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \qquad x = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad z = 5.$$

Choosing

$$r^0 = \left[\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \qquad r^1 = \left[\begin{array}{c|ccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right], \qquad r^2 = \left[\begin{array}{c|ccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right],$$

$$r^3 = \left[\begin{array}{c|ccc} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \qquad r^4 = \left[\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right],$$

$$z^0 = 0, \qquad z^1 = 39, \qquad z^2 = 2, \qquad z^3 = 2, \qquad z^4 = 2,$$

one solves four linear programs of type (A–P), arriving at a last (H–P) problem of the form

$$\min \sum \left[\begin{array}{c|ccc} \frac12 & 1-\frac12 & 1-\frac12 & 10-\frac12 \\ \frac12 & 1-\frac12 & 10-\frac12 & 1-\frac12 \\ 1 & 10-\frac12 & 1-\frac12 & 1-\frac12 \end{array}\right] (y, x) - 0$$

No solution yields $d < 0$. Hence the last solution of (A–P) is optimal:

$$\bar{r} = \left[\begin{array}{c|ccc} \frac12 & \frac12 & \frac12 & 0 \\ \frac12 & \frac12 & 0 & \frac12 \\ \frac12 & 0 & \frac12 & \frac12 \end{array}\right].$$

The integer optimum satisfies:

$$\frac52 + \sum \left[\begin{array}{c|ccc} 1 & -\frac12 & -\frac12 & 0 \\ 1 & -\frac12 & \frac12 & -\frac12 \\ 1 & \frac12 & -\frac12 & -\frac12 \end{array}\right] (y, x) \leq 0. \qquad ([illegible])$$

Conclusion. Instead of solving a LP with 12 rows and 21 variables, one solves 4 LP's with 4 rows and between 5 and 9 columns, each of them being identical with the previous one with one new column added (so that relatively few pivot steps are required). A condition (C) has been found which the integer solution must satisfy, and a bracket for the optimal value has been obtained.

$$-5 \leq z \leq 5.$$

2.2. *Benders partitioning* [6]

2.2.1 *The general scheme*

Consider the problem P:

$$\min\{z = \delta r + \gamma\chi \mid Dr + C\chi \geq \beta, r \geq 0, \chi \in S\}.$$

It can be rewritten as

$$\min_{\chi \in S}\left\{\gamma\chi + \min_{r}\{\delta r \mid Dr \geq \beta - C\chi, r \geq 0\}\right\}$$

or

$$\min_{\chi \in S}\left\{\gamma\chi + \max_{u}\{u(\beta - C\chi) \mid uD - \delta \leq 0, u \geq 0\}\right\}.$$

Let R be $\{u \mid u.D - \delta \leq 0, u \geq 0\}$. R is independent of χ. If $R = \emptyset$, P has no solution.

Otherwise, if for some χ there exists u^r, an extreme ray of R, such that $u^r.(\beta - C.\chi) > 0$, $\mathrm{D} = \max_{u \in R} u.(\beta - C.\chi)$ is unbounded and $\mathrm{L} = \min_{r \geq 0}\{\delta r \mid Dr \geq \beta - C.\chi, r \geq 0\}$ has no solution.

P can therefore be solved as G:

$$\min_{\chi \in S}\left\{z \mid z \geq \gamma\chi + \max_{u \in R}\{u(\beta - C.\chi) \mid u \text{ either extreme point of } R \right.$$
$$\left. \text{or extreme ray of } R \text{ satisfying } u(\beta - C.\chi) \leq 0\}\right\}.$$

An algorithm would proceed as follows:

Step 0. Set $k = 0$.

Step 1. Replace k by $k + 1$. Form Q, the set of indices of known extreme points u^i and extreme rays u^m of R. For $k = 1$, choose any $\chi^1 \in S$, go to 3.

Step 2. Solve G^k over Q. If G^k has no feasible solution, the same is true for P. In that case terminate. Otherwise, let χ^k, z^k be an optimal solution of G^k.

Step 3. Solve L^k and D^k with $\chi = \chi^k$. Let $f(\chi^k)$ be their optimal values. If $f(\chi^k)$ is $-\infty$, terminate with no feasible solution. If $f(\chi^k)$ is $+\infty$, there is no feasible r associated with χ^k; let u^k be the optimal extreme ray of D^k. Go to 1. If $f(\chi^k)$ is finite, let (r^k, u^k) be the optimal pair. It is a feasible solution for P. If

$$z^k \geq \gamma\chi^k + f(\chi^k),$$

then (y^k, v^k, z^k) is optimal for P: terminate. Otherwise go to 1 with a new extreme point u^k of R.

We shall call the "normal" case, in which one identifies y_1 with y, "*direct partioning*". By contrast we shall use the term "*reverse partioning*" for the identification $y_1 = x$. (It is interesting that one can, at least for some problems, bring back the integrality conditions, which in the direct case are taken care of (formally) by $y_1 \in S_1$ in an indirect fashion.)

2.2.2. Direct partioning

Starting with an arbitrary y (often $y^1 = (1, 1, \ldots, 1)$; all plants open), one solves alternatively the following two problems. Firstly L^k

$$\min\left\{\sum c(i,j)\,.\,x(i,j)\,\middle|\,x(i,j) \leq y^k(i),\ \sum_i x(i,j) = 1,\ x(i,j) \geq 0\right\}$$

and its dual D^k

$$\max\left\{\sum v(j) - \sum w(i,j)\,.\,y^k(i)\,\middle|\,w(i,j) - c(i,j) \geq v(j),\ w(i,j) \geq 0\right\}$$

with solutions $x^k(i,j)$, $v^k(j)$, $w^k(i,j)$; and secondly G^{k+1}

$$\min_{y \in S}\left\{z : z \geq \sum_i \left[f(i) - \sum_j w^h(i,j)\right].\,y(i) - \sum_j v^h(j),\ h = 1, \ldots, k\right.$$

$$\text{and } (w^h, v^h) \text{ extreme point of } R;$$

$$\sum w^h(i,j)\,.\,y(i) \geq \sum_j v^h(j),\ h = 1, \ldots, k$$

$$\left.\text{and } (w^h, v^h) \text{ extreme ray of } R\right\}.$$

Example. The 3 by 3 problem used before yields the following sequence of problems:

L: $z = 6$, inequality for G: $\sum y(i) - 3 \geq z$;
G: $z = 3$, $3 \leq z \leq 6$, $y = (0, 0, 0)$;
L: infeasible, inequality for G: $\sum_i y(i) \geq 1$;
G: $z = 4$, $4 \leq z \leq 6$, $y = (1, 0, 0)$;
L: $z = 13$, inequality for G: $(1, -8, -8)y - 12 \geq z$;
G: $z = 4$, $y = (0, 1, 0)$, $4 \leq z \leq 6$;
L: $z = 13$, inequality for G: $(-8, 1, -8)y - 12 \geq z$;
G: $z = 4$, $y = (0, 0, 1)$, $4 \leq z \leq 6$;
L: $z = 13$, inequality for G: $(-8, -8, 1)y - 12 \geq z$;
G: $z = 4.5$, $y = (.5, .5, .5)$;
L: $z = 4.5$, optimal.

We have run bigger problems and have experienced the normal difficulties towards the end, as the number of constraints in G increases. We have tried two versions of the algorithm, the one described above, and another one in which some heuristics are used to render y integer if it turns out fractional. In the second case we noticed much faster convergence. E.g., a 20 by 35 problem shows the following behavior: after 20 iterations, the first algorithm gives the interval $209.0 \le z \le 341.8$, whereas the second has arrived at $235.7 \le z \le 245$. The optimal value is 243.

2.2.3. *Reverse partitioning*

One now has to solve, let $S = \{x \mid \sum_i x(i,j) = 1, \text{ all } j\}$,

$$\min_{x \in S} \left\{ c.x + \min_y \{ f.y \mid 0 \le x(i,j) \le y(i), \forall i,j \} \right\},$$

or

$$\min_{x \in S} \left\{ c.x + \max_t \left\{ \sum_{ij} t(i,j).x(i,j) \mid t(i,j) \ge 0, \sum_j t(i,j) \le f(i) \right\} \right\}$$

i.e., $\mathbf{L}^k$ is $\min_y f.y$

$$y(i) \ge x^k(i,j), \quad \forall i,j$$

$$y(i) \ge 0, \quad \forall i$$

whose solution is clearly $y(i) = \max_j x^k(i,j)$, all i. D^k is

$$\max \sum t(i,j)x^k(i,j)$$

$$t(i,j) \ge 0, \quad \forall i,j$$

$$\sum_j t(i,j) \le f(i), \quad \forall i$$

and G^k is

$$\min_{x \in S} \Big\{ z \mid z \ge c.x + \max_t \Big\{ t.x \mid t = t^1, \ldots, t^k,$$

$$t \text{ extreme point of the set } \Big\{ t \ge 0, \sum_j t(i,j) \le f(i) \Big\} \Big\} \Big\}$$

whose dual reads

$$\max \left\{ \sum_j v(j) \mid v(j) - \sum_n w^n c_n(i,j) \le 0, \; w^n \ge 0, \; \sum_n w^n = 1 \right\}$$

with $c_n = c + t^n$. Let $d(i,j) = \sum_n w^n c_n(i,j) - v(j)$. Sum n from 1 to k. Given w, one can determine v and d via

$$v(j) = \min_i \sum_n w^n c_n(i,j) \quad \text{and} \quad d(i,j) = \sum_n w^n c_n(i,j) - v(j) \ge 0$$

(1) The reduced cost of a $d(i,j)$ is $-x(i,j)$, $0 \le x(i,j) \le 1$, so that the only candidates to enter the basis are the w's.

(2) If w^k comes in and w^h goes out, the pivot row is $\sum_k w^k = 1$.

(3) If w^k comes in and $d(i, j)$ goes out, the "pivot row" is

$$v(j) - \sum_k w^k c_k(i, j) + d(i, j) = 0.$$

(4) Consider the constraint

$$v(j) - \sum_k w^k c_k(i, j) + d(i, j) = 0.$$

(a) Either $v(j) < \sum w^k c_k(i, j)$, then $d(i, j) > 0$ is basic and the constraint gives the $d(i, j)$, or

(b) $v(j) = \sum w^k c_k(i, j)$, then

(i) either $d(i, j) = 0$ nonbasic ... row gives v or basic w,

(ii) or $d(i, j) = 0$ basic ... row gives $d(i, j)$.

For the 3 by 3 problem we obtain the continuous optimum after 10 iterations, i.e. 10 LP's with 10 rows and between 4 and 10 variables (exclusive of the slacks) of type GD. In fact one does not really need to use an LP code to solve GD, but rather one uses a specialized technique involving much less computation.

The optimal value is immediately in the interval (3, 5), then at the 2nd iteration in (4, 5), then (4th iteration) (4.167, 4.83), finally in (4.5, 4.83) at the 5th iteration. The next iterations leave the bracket unchanged, until the 10th iteration gives the optimum 4.5.

3. Modified simplex methods

3.1. *Simple plant location and the simplex method*

The SPLP lends itself rather well to solution by LP techniques, in the sense that the LP solutions are often integral. There are many bases which are unimodular (vertices which are integral). A standard simplex algorithm, however, will encounter fractional vertices.

Also, the LP tableau of the SPLP is large. E.g., a 10 plant, 20 customer problem corresponds to a LP with 20 + 10.20 = 220 rows and 10.20.2 − 10 = 410 variables, including slacks.

An efficient implementation of the simplex method, then, requires that:

(i) The structure of the LP be carried along implicitly, all relevant elements of the updated tableau being generated as needed, and

(ii) efforts be made to avoid fractional vertices.

3.2. *A triangularization algorithm* [21]

We have implemented a simple "*triangularization*" algorithm, which tries to accomplish these objectives. In outline, it functions as follows.

(1) Consider (2.1) as an equality system with slacks $s(i,j)$, i.e. with the constraints:

$$x(i,j) + s(i,j) = y(i), \quad \text{all } i, j$$

An initial triangular basis is easily found. To simplify matters, we always included (and maintained) the $y(i)$ in the basis. (In retrospect, we believe that this may be too restrictive.)

(2) At a typical iteration, given the triangularity of the basis, we can compute the dual variables by recursive scanning of the dual constraints and substitution. If the problem is not optimal (dually feasible) we select an incoming variable $t(i^*, j^*)$, which represents either $x(i^*, j^*)$ or $s(i^*, j^*)$.

(3) We generate the pivot column by scanning the primal constraints and expressing the basic variables $y^B(i)$ and $x^B(i,j)$ in terms of $t(i^*, j^*)$. This is possible on account of triangularity, and the scanning can be used to exhibit the sequence of variables which shows the triangularity of the basis explicitly. It is clear that the basic $s^B(i,j)$ can be generated afterwards from the constraints (2.1), so that only the x^B, y^B computations require iterative scanning.

(4) Given the constant column (the values of the basic variables) and the pivot column developed in (3), we can perform the standard ratio tests and decide on an outgoing variable. Let $t^B = b + t^* \cdot p + \cdots$ (p... pivot column) represent the basic variable vector t^B in terms of its current value b and the incoming variable t^*. The $b(i)$ are either 0 or 1, but the pivot column may, in general, contain integer entries other than 0, 1, -1.

In the ratio test one searches for an outgoing variable $t(i^{**}, j^{**})$ which corresponds to a maximal $b(i)/p(i)$, over $p(i) < 0$. When the maximal ratio is zero, the pivot step is *degenerate* (does not change the value of the solution; one remains at the same vertex of the polytope). It can be seen that one can then find an eligible i^{**} for which the $p(i)$ is -1, so that the new basis remains unimodular. When the maximal ratio is -1, we have a non-degenerate pivot step which leads to a new unimodular basis. When the maximal ratio is fractional, i.e., when the $p(i)$ is negative other than -1, we abandon the incoming variable t^* because the new basis would have to be non-triangular. In effect, one abandons motion along one edge of the polytope from the current vertex to what would most likely (apparently there are exceptions) be a fractional neighbor.

Comments. (i) In our code all array representations are kept in binary form. We do not generate $p(i)$ which are other than 0, 1, -1, but carry along a fourth type (represented by a code of two bits) which we designate as "*polluted*". Linear combinations of polluted entries are also designated as polluted. We abandon incoming variables (candidate edges) which lead to a polluted $p(i^{**})$. This means that our code is somewhat too restrictive (pessimistic).

(ii) The code will *fail in two cases*:

(α) There are no candidate edges leading to a unimodular new basis.

(ii) A new unimodular basis has been found and is yet non-triangular.

(iii) For very small problems (we have run a large number of problems with $m = 4$ and $n = 6$) we have not been able to get one of the two conditions mentioned above, no matter what data we tried.

Failure (a) apparently is unlikely for "easy" fixed charges. It can be "induced" most readily by using uniformly large fixed charges (rendering the problem almost fully combinatorial).

We have only one example for failure (b), for a fairly large problem of 20 plants and 35 customers. The unimodular basis which appears to be non-triangular is of size (735 by 735).

(iv) The real flaw of the method, however, lies in two other circumstances. One is the well-known problem of *degeneracy*, which leads to large numbers of apparently useless pivot steps. The other is that a method which treats the $x(i, j)$ and $s(i, j)$ as the important variables is probably doomed to failure because of *dimensionality*. We are now convinced that a direct LP technique will have to concentrate on the $y(i)$, just as is done by enumerative methods. Our choice of taking all $y(i)$ always basic and preventing them from leaving the basis was probably unwise, and the methods of the next section are probably more appropriate.

Table 1 exhibits selected computational results for small problems. The code permits slight changes in initialization and selection of incoming variables. We do not attribute any significance to such changes and only use an asterisk to distinguish between two similar yet different runs.

Table 1

Dimensions	Problem #	completed? yes	no	reason for failure	Number of iterations	Number of fractional vertices discarded	Optimal value 1st	Cont.	Value at termination
4 × 6	(1) ($f_i = 10^4$)	✓			11	0	20024	16	14
	(1) (f_i small)	✓			6	0	44		
	(2) ($f_i = 10^4$)	✓			16	1	10026	—	—
	(2) (f_i small)	✓			11	1	25	—	—
	(1*) ($f_i = 10^4$)	✓							
	(1*) (f_i small)	✓			6	0	44	—	—
10 × 10	(1) (f_i small)	✓			14	0	80	?	80
	(2) (f_i large)		✓	a	34	37	?	?	30038
	(1*)	✓			14	0	80		80
	(2*)		✓	a	55	43	?		30038
20 × 35	(1)		✓	b	54	large	242	240	252
	(1*)	✓			84	moderate	242	242	242

3.3. Some special triangular bases

The SPLP can be (as was discussed among A. Hoffman, E.L. Johnson and M. Padberg, and suggested to us by A. Hoffman) reformulated as follows in terms of variables $\bar{y}(i) = 1 - y(i)$.

$$\begin{aligned}
&\min \sum f(i)\cdot(1-\bar{y}(i)) + \sum_{i,j} c(i,j)\cdot x(i,j), \\
&\sum_i x(i,j) = 1, && \forall j, \\
&x(i,j) - 1 + \bar{y}(i) \le 0, && \forall i,j, \\
&0 \le x(i,j) && \forall i,j, \\
&y(i) \text{ in } \{0,1\}, && \forall i,
\end{aligned} \tag{3.1}$$

or

$$\begin{aligned}
&\min \left[\sum_{i,j} c(i,j)\cdot x(i,j) - \sum_i f(i)\cdot \bar{y}(i)\right] + \sum_i f(i), \\
&\sum_i x(i,j) = 1, && \forall j, \\
&x(i,j) + \bar{y}(i) + s(i,j) = 1, && \forall i,j, \\
&1 \ge x(i,j),\quad \bar{y}(i),\ s(i,j) \ge 0, && \forall i,j, \\
&y(i) \text{ in } \{0,1\}, && \forall i.
\end{aligned} \tag{3.2}$$

(3.2) is a highly structured and generally very large *set partitioning* problem. Therefore, the interesting results of [2] and [5] apply, even though their practical applicability is uncertain in view of the large problem size.

Note that:

(1) a given $x(i,j)$ appears in exactly two explicit equations, one of which $(\Sigma_i x(i,j) = 1)$ we shall term the Σj (sigma j) equation, while the other $(x(i,j) + s(i,j) + \bar{y}(i) = 1)$ shall be referred to as $*ij$ (cross ij) equation;

(2) a given $y(i)$ appears in n equations $*i1, \ldots, *in$;

(3) a slack $s(i,j)$ appears only in one $*ij$ equation. These observations are important in pointing out how basic variables can be computed. One may establish a number of useful properties:

Property 1. A basic $s(i,j)$ can be determined only from $*ij$, so that when the involved $x(i,j)$ and $y(i)$ have been determined, the basic $s(i,j)$ is known. I.e., once the basic $x(i,j)$ and $y(i)$ have been computed, the basic $s(i,j)$ can be determined in triangular fashion (one at a time). Therefore, one need only be concerned with the subbasis B^{xy}, the subbasis whose columns correspond to x and y.

Property 2. Given j, there must be at least one $x(i,j)$ basic expressed from Σj. All other basic $x(i,j)$ must come from $*ij$.

Property 3. Given i, a basic $\bar{y}(i)$ must be determined from one of the Σj. Therefore, if all $x(i,j)$ are basic, at least one of them must come from a Σj so that $\bar{y}(i)$ can be computed.

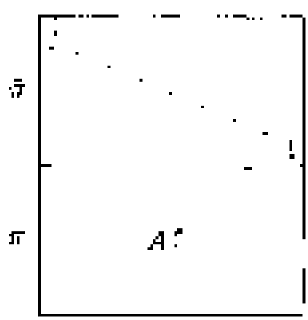
Property 4. The subbasis corresponding to basic $y(i)$ and $x(i,j)$ equal to 1 can be rearranged so as to be triangular.

The constraint matrix has only coefficients 0 and 1, the right hand side contains only 1's; there must therefore be exactly one 1 per row in the submatrix. There is no zero column. Hence there must exist a permutation of the rows and columns which brings an identity matrix to the upper part of the submatrix.

Property 5. It is always possible to complete the basis in a triangular fashion by choice of basis columns which correspond to variables at zero.

Let A be the constraint matrix; let P be the set of indices of variables at 1 and let π and $\bar{\pi}$ be suitable index sets. Then the subbasis of Property 4 is

$A^P =$



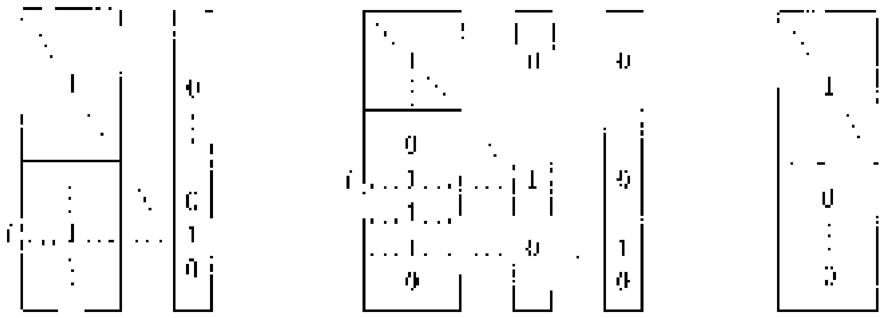
(a subscript is used for row indexing, a superscript for column indexing).

Consider A^P_π. It also has exactly one 1 per row. Consider row i, $i \in \pi$. It has one 1 in column $y(i)$, which can correspond to an x, a $\bar{y}$ or an s. We shall give one possible way of completing the basis:

(1) if the entry corresponds to an x, say $x(k,r)$, $x(k,r) = 1$ in the current solution, so that $\bar{y}(k)$ and $s(k,r)$ are 0 and not yet in A^P. Since $s(k,r)$ occurs only in one equation $(*kr)$, one can append column $s(k,r)$ to A^P.

(2) if the entry corresponds to a $\bar{y}$, say $\bar{y}(k)$. $\bar{y}(k) = 1$ means $x(k,j) = 0\ \forall j$, and $s(k,j) = 0\ \forall j$. The columns $x(k,j)$ contain two 1's and one of these might be above the main diagonal, whereas the columns $s(k,j)$ contain only one 1 and $(n-1)$ of them are adjoined to A^P with their 1's on the diagonal. One of the $s(k,r)$ will be nonbasic (its choice is arbitrary).

(3) the entry can not correspond to an $s(k,j)$ at 1, since an $s(k,j)$ column has only one 1 which is in $A^P_{\bar{\pi}}$.



$x(k,r)$ $s(k,r)$ $\bar{y}(k)$ $s(k,r_1)\cdots s(k,r_n)$ $s(k,r)$

Property 6. The basis thus constructed (which we shall call the *s-canonical* basis, since only s columns were added), has the *anti-involutive* property:

$$B=\begin{array}{c} \\ \bar{\pi} \\ \pi \end{array}\begin{array}{c}\begin{array}{cc} p & \bar{p} \end{array}\\ \left[\begin{array}{c|c} I & 0 \\ \hline A^{\xi}_{\pi} & I \end{array}\right]\end{array}, \qquad B^{-1}=\begin{array}{c} \\ p \\ \bar{p} \end{array}\begin{array}{c}\begin{array}{cc} \bar{\pi} & \pi \end{array}\\ \left[\begin{array}{c|c} I & 0 \\ \hline -A^{\xi}_{\pi} & I \end{array}\right]\end{array}$$

Property 7. The top rows $\bar{\pi}$ of the *updated tableau* are unchanged, whereas the "*bottom*" rows π are equal to the original rows minus one of the top rows.

Let $T=B^{-1}\cdot A$, then

$$T=\begin{array}{c} \\ p \\ \bar{p} \end{array}\begin{array}{c}\begin{array}{cc} \bar{\pi} & \pi \end{array}\\ \left[\begin{array}{c|c} I & 0 \\ \hline -A^{\xi}_{\pi} & I \end{array}\right]\end{array} \times \begin{array}{c} \\ \bar{\pi} \\ \pi \end{array}\begin{array}{c}\begin{array}{cc} B & N \end{array}\\ \left[\begin{array}{c|c|c} I & 0 & A^{N}_{\bar{\pi}} \\ \hline A^{\xi}_{\pi} & I & A^{N}_{\pi} \end{array}\right]\end{array} = \begin{array}{c}\begin{array}{cc} B & N \end{array}\\ \left[\begin{array}{c|c} I & A^{N}_{\bar{\pi}} \\ \hline & A^{N}_{\pi}-A^{\xi}_{\pi}A^{N}_{\bar{\pi}} \end{array}\right]\end{array}$$

and since A^{ξ}_{π} has only one nonzero element per row, one subtracts one row of the top from one row of the bottom.

More precisely, let us use the following notation:

(a) in each column j, let $i(j)$ be the route on which x is 1:

$$x(i(j), j) = 1.$$

(b) if $\bar{y}(i) = 1$, $(n-1)$ of the $s(i, j)$ are basic. Let ij_0 be the index of the nonbasic s (notice that $x(i, j)$ is also nonbasic); we shall refer to $(x(i, j_0), s(i, j_0))$ as the *nonbasic pair* associated with a $\bar{y}(i)$ at 1 (i.e., with a closed plant), then

$$T^{x}_{ij_0} = 1,$$

$$T^{s_{ij_0}}_{x_{i(j)j}} = 1, \quad T^{s_{ij_0}}_{y_i} = 1, \quad T^{s_{ij_0}}_{s_{i(j)j}} = -1, \quad T^{s_{ij_0}}_{s_{ij}} = -1, \quad j \neq j_0;$$

$$T^{x_{ij_0}}_{x_{i(j_0)}} = 1, \quad T^{x_{ij_0}}_{y_i} = 1, \quad T^{x_{ij_0}}_{s_{i(j_0)}} = -1 \quad \text{if } s_{ij} \in B,$$

$$T^{x_{ij'}}_{x_{i(j')}} = 1, \quad T^{x_{ij'}}_{s_{ij'}} = -1, \quad j' \neq j_0,$$

are the nonzero nonbasic entries of the updated tableau.

In the y *canonical* basis no $\bar{y}(i)$ or $x(i, j)$ at 0 is basic.

Property 8. The reduced costs of the nonbasic variables are:

$$d^N = c^N - c^B \bar{A};$$

in particular

$$d(\bar{y}(k)) = -f(k),$$

$$d(x(i, j)) = c(i, j) - c(i(j), j) + (s(i, j) \in N) \cdot f(i),$$

$$d(s(i, j)) = f(i),$$

where

$$(s(i, j) \in N) = \begin{cases} 1 & \text{if } s(i, j) \in N, \\ 0 & \text{if } s(i, j) \in B. \end{cases}$$

Every nonbasic $\bar{y}(k)$ is therefore a candidate for entering the basis; a nonbasic $x(i, j)$ is a candidate if its cost plus possibly the ith fixed charge is smaller than the cost of the route currently used in column j. All these pivot steps are degenerate, since the bottom part of the right hand side consists of 0's, and each candidate column has positive entries in the bottom part. Any move to a better neighbor integer vertex therefore involves a block pivot (see [2] and [3]).

Example. Take f and c as in 2.1. Consider the solution

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The cost is 13. One can construct the s-canonical basis displayed in Table 2, making $s(2,2)$ and $s(3,1)$ nonbasic on account of the large associated costs of x.

Table 2. s-canonical basis [illegible]

	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]													1			1					
2		1												1			1				
3			1															1			
[illegible]																			1		
4																1				1	
[illegible]						1							1			1					
2							1										1				
3								1										−1			
21				1					1				1	1					1		
22										1				−1					−1		
23											1					−1				1	
24												1				−1		1		1	

$d^T = (0, 10, -9, 10, 0, \ldots 9, 1, 1, -1)$

The negative entries in the nonbasic tableau belong to the updated tableau, the positive ones are the original entries which are preserved in the transformation to the updated tableau. $x(2,3)$, $x(3,5)$, $y(1)$ are candidates to enter the basis, but all yield degenerate steps if taken alone.

Block pivot [3]. One looks for a set K of nonbasic columns to bring into the basis at level 1, such that:

$$\sum_{j \in K} T_i^j = 0 \text{ or } 1 \quad \text{if } i \in P$$

(giving the ith basic variable value 1 or 0),

$$\sum_{j \in K} T_i^j = 0 \text{ or } -1 \quad \text{if } i \in \bar{P},$$

(rendering the ith basic variable 0 or 1).

For instance, bringing in $x(2,3)$ at level 1 saves 9, renders $s(2,3)$ infeasible (= −1), which has to be corrected by bringing in at 1 either $s(3,3)$ (costs $10-9=1$) or $s(2,2)$ (saves $9-1=8$). Both changes render the problem feasible, therefore yield neighbor vertices. Choosing the improving vertex, we get $x(2,3) = s(2,2) = 1$ and

$$y = \begin{bmatrix}1\\1\\0\end{bmatrix},\qquad x = \begin{bmatrix}1&1&0\\0&0&1\\0&0&0\end{bmatrix},\qquad s = \begin{bmatrix}0&0&1\\1&1&0\\0&0&0\end{bmatrix},\qquad \bar{y} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

with a cost of $13 - 8 = 5$.

Property 9. Given the set P of variables at 1, one can also complete the basis by first adding columns corresponding to $\bar{y}(i)$ at zero, then only adding s columns as needed.

We shall call this basis the *$y - s$ canonical basis*. Essentially, the procedure is the following: bring the identity matrix to the top of the basis as before, which corresponds to placing first the rows $\Sigma_1, \ldots, \Sigma_n$, then for each $\bar{y}(i)$ at 1 ($y(i) = 0$, $x(i,j) = s(i,j) = 0\ \forall j$) choose a nonbasic pair $(i, j_0(i))$ (or j_0 when not ambiguous) as before. Then, for each $\bar{y}(i)$ at 0 ($y(i) = 1$, the plant is open) some $x(i,j)$ must be 1 in a basic solution since not all $s(i,j)$ can be simultaneously positive and thus basic, as $\bar{y}(i)$ also must come from one $*ij$. Choose one index j for which $x(i,j)$ is 1 and render $s(i,j)$ nonbasic. Then complete with columns corresponding to $x(i,j) = 1$ (possible only with $\bar{y}(i) = 0$ basic and $x(i,j)$ nonbasic). This is still a triangular procedure. Finally complete with $s(i,j) = 0$.

Table 3: (B, T^{b}).

Σ1	1												1					[illegible]			
2																			[illegible]	[illegible]	
3			1														1				[illegible]
*31				1									1	1							
11	1				1								−1*		1			−1*			
23			1			1										1	1*				1*
13							1						1*		−1*		1	1*			
21						1		1								1*	1*	1			2*
22						1			1							−1*	1*		1		1*
12		1								1			1*		−1*			1*	−1*	−1*	
32				1							1		1*	1*						[illegible]	
33				1								1	1*	−1*							1

Example (cont.) The starred entries in Table 3 belong to the updated tableau.

$$d^Y = [9\ 1\ 1\ 1\ 8\ \ \ 1\ 9\ 0\ \ -1].$$

Two columns are candidates to enter the basis: $x21$ and $x33$ (in slightly simplified notation) at a saving of 1, but both pivot steps would be degenerate.

If one brings in $x21$ at level 1, $s12$ becomes -1, which can be corrected by bringing in at 1:

$s11$ at no improvement, but this is a feasible neighbor vertex.

$x22$ at a cost of $9-1=8$, which is also feasible, so that there is no need to pursue this combination further.

$x32$ at a saving of 1, but this renders $s32$ infeasible, which can be corrected by setting $x31$ or $s31$ at 1, with no improvement left. $s31$ would yield a feasible point, $x31$ would create an infeasibility which could not be corrected at a saving.

If one brings $x33$ in at 1, $s33$ becomes -1. $s31$ would completely correct it at no saving, $x31$ would cost $9-1=8$ and no further saving is possible. The solution is therefore *optimal* in integer variables.

Property 10. Given an integer feasible solution to SPLP, one can also define a triangular basis (so-called *$y-x-y-s$ canonical basis*) having the following columns: all $\bar{y}(i)$'s at 1, some $x(i,j)$'s at 0 for closed plants, all $x(i,j)$'s at 1, all $\bar{y}(i)$'s at 0, all $s(i,j)$'s at 1, some $s(i,j)$'s at 0.

We suggest that constructing such a basis as follows, one may achieve the goal of arriving at a relatively small number of negative reduced costs (i.e., to position oneself in a sense close to an optimal solution, in order to finish via relatively few and simple block pivots).

(1) Place first the columns corresponding to $\bar{y}(i)=1$ $(x(i,j)=s(i,j)=0\ \forall j)$ and associate with each i a row $*(i,j_0)$, where $(x(i,j_0(i)), s(i,j_0(i)))$ will be a nonbasic pair such that $f(i)+c(i,j)-f(i(j))-c(i(j),j)$ is maximal for $j=j_0(i)$.

(2) For each i with $\bar{y}(i)=1$, for each pair $(x(i,j), s(i,j))$, $j\neq j_0$, if $c(i,j)\leq f(i(j))+c(i(j),j)$ make $x(i,j)$ nonbasic, otherwise make $s(i,j)$ nonbasic and add row $*ij$ and column $x(i,j)$ to the subbasis.

(3) add rows $\sum j$ and columns $x(i(j),j)$.

(4) For $y(i)=0$ $(y(i)=1$, one $x(i,j)$ at least is 1), choose one $j_1(i)$ such that (a) $x(i,j_1)=1$ and (b) the increase in cost for shipping from another plant is maximal over $\{j \mid x(i,j)=1\}$ for $j=j_1$. Note that for different i, we'll get different $j_1(i)$ as there is only one $x(i,j)$ at 1 per column. We can therefore talk of $j_1^{-1}(j)$ with the convention that $f(j_1^{-1}(j))$ is $f(i)$ if $j=j_1(i)$ and is 0 if there is no i such that $j=j_1(i)$. Then add row $*ij_1(i)$ and column $\bar{y}(i)$ to the submatrix.

(5) Complete with the slacks at 1, and then some slacks at 0. Then, if we call $B_s(i)=\{j \mid x(i,j)$ is basic, $s(i,j)=0\}$, we have the following properties:

$$d(s_{ij_1(i)})=f_i-c_{ij_1(i)}-\sum_{j\in B_s(i)} f_{i(j)}+\sum_{j\in B_s(i)} (c_{ij}-c_{i(j)j}),$$

$$d(x_{ij_1(i)})=c_{ij_1}+d(s_{ij_1(i)}),$$

for $j\in B_s(i)$, $d(s_{ij})=f_{i(j)}-c_{ij}-c_{i(j)j}$,

for $x(i,j)$ nonbasic with $s(i,j)=1$, $d(x_{ij})=c_{ij}-[c_{i(j)j}+f_{i(j)}]$,

$$d(s_{ij_0(i)})=f_i\quad \text{(all other } s(i,j) \text{ are basic)}$$

Table 4. Original basis (in triangular form).

	$\bar{y}_1$	$\bar{y}_4$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	$\bar{y}_1$	$\bar{y}_3$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
15	1																													
45		1																												
32	1		1																											
41		1		1																										
42		1			1																									
44		1				1																								
11				1			1																							
12			1		1			1																						
13									1																					
14						1				1																				
25											1																			
36												1																		
12								1					1																	
31							1							1																
11													1		1															
17														1		1														
[illegible]													1				1													
24														1				1												
25													1						1											
26													1																	
14										1			1																	
21	1																					1								
22	1																						1							
24	1																							1						
35	1																								1					
32									1					1												1				
35											1			1													1			
36												1		1														1		
45		1																											1	
46		1																												1

Example.

$$r = \begin{bmatrix} 5 \\ 10 \\ 10 \\ 10 \end{bmatrix}, \qquad c = \begin{bmatrix} 3 & 2 & 4 & 2 & 20 & 5 \\ L & 1 & 3 & L & 1 & L \\ 1 & L & 1 & L & 3 & 2 \\ 5 & 5 & 5 & 1 & 3 & 4 \end{bmatrix}, \quad x = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & . & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

L stands for a very large number. Tables 4 and 5 display the original basis and updated nonbasis.

Table 5. Updated nonbasis (asterisked entries introduced in inversion process).

Basis	[illegible]
[illegible]	[illegible]

4. Enumeration

The SPLP behaves relatively well under enumeration. Enumerative codes often start with all plants open, so that a "forward step" in the search consists of closing a plant. (E.g., see [24].) More generally, one starts at a point with some plants fixed open ($i \in E$, $y(i) = 1$), some fixed closed ($i \in C$, $y(i) = 0$), some free but tentatively open ($i \in F1$, $y(i) = 1$), some free but tentatively closed ($i \in F2$, $y(i) = 0$).

4.1. State enumeration [20]

By the "state" at node n, we mean a partitioning of the index set into $E, C, F1, F2$. With the state one associates a solution (z^*, y^*, x^*) by

$$
\begin{aligned}
& y^*(i) = 1 \quad \text{for } i \in E \cup F1, \\
& y^*(i) = 0 \quad \text{for } i \in C \cup F2, \\
& z^* = \sum c(i(j), j), \\
& x(i(j), j) = 1, \\
& x(i, j) = 0, \qquad i \neq i(j), \\
& i(j){:}\ c(i(j), j) = \min\{c(i, j)\} \text{ over } E \cup F1.
\end{aligned}
\tag{4.1}
$$

Consider the problem SPLPD dual to (2.1):

$$
\begin{aligned}
& \max z_D = -\sum w(i, j) - \sum v(j), \\
& c(i, j) + w(i, j) - v(j) \geq 0, \\
& w(i, j),\ v(j) \geq 0, \\
& i \text{ over } E' \text{ and } F1', \quad j \text{ over } J = 1, 2, \ldots, n.
\end{aligned}
\tag{4.2}
$$

In terms of z and z^* (a known upper bound on z), one has the Benders inequality:

$$
\sum_{F1}\left(-f(i) - \sum w(i, j)\right) \cdot y(i) + \sum_{F2}\left(f(i) - \sum w(i, j)\right) \cdot y(i) \geq z^* - z. \tag{4.3}
$$

In [20, 24], the coefficients of the $y(i)$ (multiplied by -1) were called "global gain functions", $g(i)$, and play a central role in curtailing and guiding the search.

4.2. Strengthening the gain functions

It is important to have the $g(i)$ as small as posible. E.g., $g(i) \leq 0$ permits the fixing of $y(i)$.

It is interesting that there is a great latitude in choosing the $w(i, j)$ of (4.2), and with them the $g(i)$. One good and not completely obvious choice of the dual variables is: Let

$$c(\bar{i}(j), j) = \min\{c(i,j) \mid i \in I^{+} - I^{-} - \{i(j)\}\},$$

$$v^*(j) = \begin{cases} c(\bar{i}(j), j) & \text{if } c(\bar{i}(j), j) \geq c(i(j), j) \\ c(i(j), j) & \text{otherwise,} \end{cases} \tag{4.4}$$

$$w^*(i,j) = \max\{0, v(j) - c(i,j)\}.$$

Then one has, correspondingly:

$$\begin{aligned} g(i) &= f(i) - \sum_{j: i = i(j)} \max[0, c(\bar{i}(j), j) - c(i(j), j)], \quad i \in F1, \\ g(i) &= -f(i) + \sum_{j} \max\{0, c(\bar{i}(j), j) - c(i,j)\}, \quad i \in F2. \end{aligned} \tag{4.5}$$

As in [24], one can also define "local" gain functions to aid in both curtailment of search and strategy.

Now, it is quite clear that the reduced costs of any LP tableau associated with the variables of a state problem at node ν have the properties of gain functions. There is then substantial interest in generating the various bases of Section 3 for node ν, and utilizing the reduced costs in the enumeration.

More fundamentally, it is clear that the gain functions as used in the past only exploit a limited portion of the updated LP tableau. Having the entire updated tableau at one's disposal, at a cost which is relatively modest given the nice properties of "canonical bases", should permit substantial improvements.

In a sense, it provides a grasp of the LP polytope, e.g. by defining the edges leading away from the state point. With some ingenuity, an improvement of enumerative procedures should be attainable.

References

[1] D.A. Babayev, Comments on a Note of Frieze, Math. Programming 7 (1974) 249–252.

[2] E. Balas and M.W. Padberg, On the set-covering problem, J. ORSA, 20, (1972) 1152–1161.

[3] E. Balas and M.W. Padberg, On the set-covering problem II. An algorithm, Mgmt. Sciences Research Rep. 295, Carnegie Mellon Univ., 1972.

[4] M.L. Balinski, Fixed cost transportation problems, Naval Res. Logistics Quarterly, 8 (1961) 41–54.

[5] M.L. Balinski and K. Spielberg, Methods for integer programming: Algebraic, combinatorial, and enumerative, in J. Aronofsky, ed., Progress in Operation Res., Vol. 3, (Wiley, New York, 1969) pp. 195–292.

[6] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Num. Math., 4 (1962) 238–252.

[7] O. Bilde and J. Krarup, Sharp lower bounds and efficient algorithms for the simple plant location problem, Research Report #75/5, Inst. of Datalogy, Univ. of Copenhagen, 1975 (based on report in Danish of 1967).

[8] P. Broise, P. Huard and J. Sentenac, Decomposition des Programmes Mathematiques (Dunod, Paris, 1968).

[9] V.P. Cherenin, Reshenie nechotorych combinatornych zadach optimalnogo planirovania metodom posledovatelnych raschetov (Solving some combinatorial problems of optimal planning by the method of successive calculation), in *Proc.* of the conference of experiences and perspectives of the application of mathematical methods and electronic computers in planning, Mimeograph, Novosibirsk, 1962.
[10] G. Cornuejols, M.L. Fischer and G.L. Nemhauser, An analysis of heuristics and relaxations for the uncapacitated location problem, Cornell Univ., Techn. Report #271, Aug. 1975.
[11] G.B. Dantzig and Ph. Wolfe, Principle of decomposition for linear programs, J. ORSA, 8 (1960) 101–111.
[12] P.S. Davis and T.L. Ray, A branch-bound algorithm for the capacitated facilities location problem, Naval Res. Log. Quart., 16 (1969) 331–344.
[13] M.A. Efroymson and T.L. Ray, A branch-bound algorithm for plant location, J. ORSA, 14 (1966) 361–368.
[14] A.M. Frieze, A cost function property for plant location problems, Math. Programming, 7 (1974) 245–248.
[15] A.M. Geoffrion, Lagrangean relaxation for integer programming, Math. Programming Study 2 (1974) 82–114.
[16] A.M. Geoffrion and G.W. Graves, Multicommodity distribution system design by Benders decomposition, Management Sci., 20 (1974) 822–844.
[17] F. Glover, Compact LP bases for a class of IP problems, Report #75-18, Mgmt. Sc. Report Series, Grad. School of Bus. Adm., Univ. of Col., April 1975.
[18] R.S. Garfinkel, A.W. Neebe and M.R. Rao, An algorithm for the M-median plant location problem, Working Paper #7312, Grad. School of Management, Univ. of Rochester, May 1973.
[19] G.W. Graves and R.D. McBride, The factorization approach to large-scale linear programming, Working Paper #208, Western Mgmt. Sc. Inst., UCLA, Aug. 1973.
[20] M. Guignard and K. Spielberg, Search techniques with adaptive features for certain integer and mixed-integer programming problems in: A.J.H. Morrell, ed., Information Processing 68, Volume 1 (North Holland, Amsterdam, 1969) pp. 238–244.
[21] M. Guignard and K. Spielberg, Triangular structures in uncapacitated plant location, ORSA/TIMS Meeting, 1974, San Juan, Puerto Rico.
[22] R.E. Marsten, An algorithm for finding almost all of the medians of a network, Report, Center for Math. Studies in Econ. and Mgmt. Sc., Northwestern Univ., Nov. 1972.
[23] L. Schrage, Implicit representation of generalized variable upper bounds in linear programs, Report #7543, Dept. Econ. & Grad. School of Business, Univ. Chicago, Oct. 1975.
[24] K. Spielberg, Plant location with generalized search origin, Management Sci., 16 (1969) 165–178.
[25] K. Spielberg, Algorithms for the simple plant-location problem with some side conditions, J. ORSA, (17) (1969) 85–111 (based on IBM New York Sci. Center Report #320-2900, May 1967).
[26] H.P. Williams, Experiments in the formulation of integer programming problems, Math. Programming Study 2 (1974) 180–197.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 279–284

REDUCTION METHODS FOR STATE ENUMERATION INTEGER PROGRAMMING

Monique GUIGNARD
Department of Statistics, Wharton School, University of Pennsylvania, Philadelphia, PA 19174, U.S.A.

Kurt SPIELBERG
Scientific Marketing, IBM, White Plains, NY 10604, U.S.A.

Integer programs with small bound intervals can often be dealt with effectively by a state enumeration procedure with reduction methods. Our approach features the consistent use of logical inequalities derived during the computation, especially for influencing the choice of appropriate directions for the search effort.

1. Introduction

In spite of much good research and a host of proposed algorithms, the all-integer program P

$$\begin{aligned} \min\ & c \cdot y = z \\ & C \cdot y \leq b \qquad\qquad (1.1) \\ & y(j) \text{ in } [L(j), U(j)], \quad j = 1, 2, \ldots, n \\ & y(j) \text{ --- integer,} \end{aligned}$$

is far from being solved successfully even for small problems.

Both branch-and-bound (BB) programming (see [4] for a recent survey) and enumerative programming meet success for some problems (usually those with which the analyst is familiar) and fail badly elsewhere.

There is a need for a flexible integer programming system, possibly with user intervention on some kind of interactive level. In this paper we discuss an experimental enumerative system which is meant to incorporate a family of techniques which we have shown, or which we believe, to have substantial promise.

Among the practical problems which may require such techniques, we cite large scale integer problems with substantial logical structure. Many of these are scheduling problems with time dependent (0, 1) decision variables, say $y(i, t)$. (E.g., $y(i, t)$ might be 1 (0) if a certain choice is made (not made) in time period t.)

A production code of the BB type (such as MPSX/MIP of IBM) takes a good deal of time solving linear programs. A promising alternative, then, is to solve only one linear program (at level 0 of the search), or conceivably a linear program whenever the search returns to level 0, and to finish by state enumeration.

MPSX/MIP 370 has a control language which would allow the writing of an enumeration program (using procedures of MPSX) in **PL/I**. Knowledge about special structure can probably be incorporated best in such an enumerative code.

Another area of interest for the techniques of this paper is to be found *within* a production mixed-integer code, especially for problems with a large number of continuous variables. The reduction and state enumeration procedures over Benders inequalities would be executed entirely in core storage and would consume negligible effort compared to the other I/O-bound solution procedures.

2. State enumeration

2.1 Scheme of search

The search is organized as follows. It starts at "level $l = 0$" and "node $\nu = 1$", with all components "free", i.e., with all components $y(j)$ only constrained by the initial bounds $L(j)'$ and $U(j)'$.

At a general iteration, one is at *level* l which measures the number of explicit bound changes ("forward branches") which have been imposed from the last time the search was at level 0.

At level l (and node ν; ν is a running counter, increased by one at each iteration), one basically takes one of two actions:

(i) A *Forward Step* from level l to level $l+1$ (setting the bound of a branch variable to a new value).

(ii) A *Backward Step* from level l to level $l-1$ (with the search terminating when $l-1$ is -1).

The explicit forward steps from level 0 to level l are recorded in two lists of l numbers:

List 1 consists of signed component indices, the sign of an index reserved for indicating whether the associated variable was constrained by a raising (lowering) of its lower (upper) bound.

List 2 contains the value to which the upper (lower) bound of the related component from list 1 is to be lowered (raised) on *return* to a level.

It is clear that such a scheme suffices to record the history of the search and to control the search on backward steps. Further details can be skipped.

2.2 The state

At node ν (level l) one easily computes a set of "Working Bounds", $\{L(j), U(j)\}$ i.e., a set of bounds determined by the explicit branches of the search.

as well as by subsequent applications of the reduction procedures. (We shall usually drop the superscript ν on the working bounds.)

The *State* S^{ν} is essentially meant to be a *conjectured value* y^{ν}, for which we permit (to keep the search simple) the choice of setting a given $y^{\nu}(j)$ either to its lower bound $L(j)$ or to its upper bound $U(j)$.

It is the state value y^{ν} which is substituted as a trial solution. Forward branches are taken so as to lead away from the state, and are chosen among some index set J (see Section 4.2) so as to reduce total infeasibility.

In the numerical experimentation we determined an initial state at node $\nu = 1$ from the initial LP solution y':

$$
\begin{aligned}
y^{1}(j) &= L(j) \quad \text{if} \quad y'(j) \leq L(j) + r\cdot(U(j) - L(j)) \\
&= U(j) \quad \text{if} \quad y'(j) > L(j) + r\cdot(U(j) - L(j))
\end{aligned}
\tag{2.1}
$$

(r being an arbitrary rounding parameter). At subsequent levels the state is carried along, i.e. one uses the transformed state given by:

$$
\begin{aligned}
y^{\nu}(j) &= L(j) \quad \text{if} \quad y^{\nu-1}(j) \quad \text{was} \quad L(j)' \\
&= U(j) \quad \text{if} \quad y^{\nu-1}(j) \quad \text{was} \quad U(j)'.
\end{aligned}
$$

Alternatively, we also considered the options of setting $y^{\nu}(j)$ always to the lower working bound (in the tables indicated by "ALWL") or always to the upper working bound (in the tables: "ALWU").

3. Techniques for integer state enumeration

No one technique can be expected to solve all problems. A modular collection of techniques, possibly controlled in an interactive fashion, may eventually prove to be the best vehicle for studying and resolving general and special integer programs.

In Section 3.1 below, we outline those techniques which will be stressed in this paper, and for which some numerical results will be given. In Section 3.2 we outline other methods which we have tried and for which results have been given elsewhere. Some of these methods need to be generalized from the 0–1 to the integer programming case.

The results of this paper demonstrate (for small problems; but we believe that there is no reason to assume drastically different behavior for larger problems) the importance of *state enumeration* (good starting points for the enumeration) and of some form of *reduction* (i.e., systematic tightening of bounds).

The generation of logical ("preferred") inequalities does not yield, in these experiments, much additional improvement. We believe that this shows the necessity of combining such techniques with the use of penalties and propagation (see Section 3.2, items 7 and 8).

3.1 Techniques used in current experimentation

(1) *Solution of only one linear program* at the start (possibly followed by judicious use of cutting planes to get optimal but non-integral tableaux with a relatively good value for z, i.e. a large value for the objective function of the relaxed problem).

(2) *Retention of the top row* (or the related Benders inequality, [2]) for purposes of bound reduction or fixing of variables. For some purposes one may wish to retain the entire tableau. Suitably updated they are referred to as "*current*" top row or tableau.

(3) Definition and use of a *state*, i.e., a suitable origin for the search (see [8] or [9]) (either *permanently* after solution of the initial LP, or *dynamically* according to some (heuristic) criterion at each *level* of the search).

(4) "*Reduction*" *of system* (1.1) at *every level* of the search

(a) Reduction of the *bound intervals* for the $y(j)$ and the slacks $s(i)$ (as proposed by Zionts [17]).

(b) *Construction of logical relations* ("minimal preferred inequalities") which are to guide the search so as to: (i) find feasible solutions, (ii) "minimize" the search effort.

Each preferred inequality specifies d (degree) *preferred* or *indicated* bound changes. One indicated bound change, at least, must be implemented if the problem is to have a solution.

The main emphasis is on "*contraction*", i.e., on guiding the search into (locally) increasingly constrained directions [6, 14, 16].

(5) *Local search* of lattice points close to a given point y. A simple procedure for looking at all points which differ from y in exactly "lev" (level: 1-level, 2-level search) components by exactly one unit. The search is also used as a strategic device, to select branches for getting to new points with decreased overall infeasibility. As can be seen, there is some overlap and conflict between (4) and (5) (see also Section 5).

3.2 Techniques to be incorporated in a full system

(6) *Cutting plane* techniques. Our experimental system includes the ability of adding cuts, followed by reoptimization (see [5, 10, 11]). The (0, 1) test problems of this paper can be solved by such cutting plane methods, with only little enumeration. More difficult problems (with larger gap between LP and IP objective function) may prove intractable.

(7) *Penalties* and preferred variable inequalities. Preferred variable inequalities (as this paper shows in conjunction with [7, 15, 16]) are best invoked together with penalties. One rules out certain branches of a suitable preferred inequality due to large associated penalties and pursues alternatives when they are favorable from a "contraction" point of view.

(8) *Propagation.* An "indicated branch (bound change)" of a minimal preferred inequality (see Section 4.2) may often be implemented by "propagation", tantamount to fixing a variable (or altering a bound) at its current values, i.e. at insignificant computational cost. See [15] for some excellent computational results.

(9) *Mixed* integer problems. For a mixed problem, one may work with Benders inequalities generated during the search. Techniques (1)–(8) can then be applied to a system of Benders inequalities in the integer variables, which in our experience lend themselves well to reduction.

4. Reduction for integer variables

4.1 *Reduction of bound intervals* [17]

Consider the constraint set of problem (1.1) in equality form:

$$A \cdot x = b \tag{4.1}$$
$$L(j) \le x(j) \le U(j), \quad j = 1, 2, \ldots, n + m$$

with x a composite of the structural variables $y(j)$ $(j = 1, 2, \ldots, n)$ and the slacks $s(i)$ $(i = 1, 2, \ldots, m)$.

From (4.1) a new set of bounds can be computed in accordance with the formulas:

$$L(j)' = \max \begin{cases} L(j);\ U(j) - (b(i)/a(i,j)) - (1/a(i,j))(\mathrm{APU}(i) + \mathrm{AML}(i)) \\ \quad \text{for } i : a(i,j) > 0, \\ U(j) + (b(i)/a(i,j)) - (1/a(i,j))(\mathrm{APL}(i) + \mathrm{AMU}(i)) \\ \quad \text{for } i : a(i,j) < 0; \end{cases}$$

$$U(j)' = \min \begin{cases} U(j);\ L(j) + (b(i)/a(i,j)) - (1/a(i,j))(\mathrm{APL}(i) - \mathrm{AMU}(i)) \\ \quad \text{for } i : a(i,j) > 0, \\ L(j) - (b(i)/a(i,j)) - (1/a(i,j))(\mathrm{APU}(i) + \mathrm{AML}(i)) \\ \quad \text{for } i : a(i,j) < 0; \end{cases}$$

$$j = 1, 2, \ldots, n + m. \tag{4.2}$$

$$\begin{aligned} \mathrm{APU}(i) &= \sum a^{+}(i,j) \cdot U(j) \\ \mathrm{APL}(i) &= \sum a^{+}(i,j) \cdot L(j) \\ \mathrm{AMU}(i) &= \sum a^{-}(i,j) \cdot U(j) \\ \mathrm{AML}(i) &= \sum a^{-}(i,j) \cdot L(j). \end{aligned} \tag{4.3}$$

Summation is from 1 to $n + m$.

$$a^+(i,j) = \max(0, a(i,j)),$$
$$a^-(i,j) = \min(0, a(i,j)). \tag{4.4}$$

These formulas are easily derived. They are slightly altered from some of the formulas found in [17].

At any node of the enumeration one applies (4.2) iteratively until there is no more alteration of bounds. It appears to us (after some experimentation) that this procedure is somewhat preferable to the equivalent one of Section 4.2 below (which gives the same results by iterative application of minimal inequalities of degree 1).

In the numerical experiments which we conducted, little use was made of the resulting bounds on the slacks. They could be exploited, for example, in Section 4.2.

4.2. Logical inequalities for integer variables

(i) *The* (0,1) *case* [1, 6, 14, 16]. Let (1.1) be a system in zero-one variables. One can then associate with it a "minimal preferred variable" system

$$Q \cdot y \geq q \tag{4.5}$$

of degree d. (The unusual case that (4.5) is empty, i.e. that (1.1) does not imply any logical relations for the $y(j)$, can be taken care of by simple default procedures. In the following, it is always assumed that (4.5) is not empty.)

Each row k of Q has d non-zero entries and row k of (4.5) represents one logical condition implied by the system and the zero-one conditions. Let $q(k,j)$ be the entries in row k of Q.

$q(k,j) = 1$ (-1) implies that the possibility $y(j) = 1$ (respectively $y(j) = 0$) should be considered as a logical alternative (i.e., as *preferred* or *indicated* value) in an either-or partitioning. E.g., $d = 3$, and $q(k,j1) = -1$, $q(k,j2) = 1$, $q(k,j3) = 1$, $q(k,j) = 0$ represents the logical implication:

either $y(j1) = 1$, or $y(j2) = 0$, or $y(j3) = 1$.

(ii) *The integer case*

(a) *Reduction*. Starting with (1.1), one multiplies the columns of C by the bound intervals $U(j) - L(j)$, and correspondingly subtracts $\sum c(i,j) \cdot L(j)$ from $b(i)$, for each i. In this fashion one effectively changes to a system with variables $r(j)$ in the unit hypercube, i.e. to

$$\sum d(i,j) \cdot r(j) \geq b'(i), \tag{4.6}$$

$$r(j) = (y(j) - L(j))/(U(j) - L(j)),$$

$$d(i,j) = c(i,j) \cdot (U(j) - L(j)),$$

$$b'(i) = b(i) - \sum L(j) \cdot c(i,j),$$

$$0 \leq r(j) \leq 1.$$

The procedures for generating all minimal preferred inequalities for zero-one variables [16] are then applied to (4.6) as if (4.6) were a system in (0, 1) variables.

When the $t(j)$ are true zero-one variables, then an inequality such as

$$2 \cdot t(j1) - 3 \cdot \bar{t}(j2) \geq -1$$

(with $\bar{t}(j) = 1 - t(j)$) is interpreted as

$$t(j1) + t(j2) \geq 1,$$

i.e., either $t(j1) = 1$ or $t(j2) = 0$.

In the case of (4.6), however, all one can imply is that either the lower bound of $y(j1)$ must be raised, or the upper bound of $y(j2)$ must be lowered.

The minimal preferred inequalities (obtained by exactly the same procedures as for zero-one variables) are best written as:

$$\sum_{\pi} \tilde{t}(j) \geq 0,$$

$$\tilde{t}(j) = t(j) \quad \text{or} \quad \tilde{t}(j) = 1 - t(j). \tag{4.7}$$

The degree d is equal to $|\pi|$, the cardinality of the preferred set under consideration.

It should be noted that a partitioning relation such as (4.7) is usually much stronger than conventional branch-bound dichotomies. In a branch-bound code, the partitioning of the above example, for instance, could be exploited by the successive solution of the two problems $P[t(j1) = 1]$ and $P[t(j2) = 0$ *and* $t(j1) = 0]$. Clearly, analogous conjunctive conditions are imposed in integer branch bound programming. See [15, revised] for some details.

In enumerative programming, such conjunctive conditions are taken care of automatically by the book-keeping.

One may summarize the situation more generally. Let

$$v(j) = L(j) \quad \text{if} \quad \tilde{t}(j) = t(j)$$
$$\qquad U(j) \quad \text{if} \quad \tilde{t}(j) = 1 - t(j), \quad \text{all} \quad j \in \pi.$$

Theorem 1. *In order that y be an integer solution of (1.1), it is necessary that $\tilde{t}(\pi) \neq 0$, i.e. that $y(\pi) \neq v(\pi)$.*

Let $\tilde{y}(j) = y(j) - L(j)$ (resp. $U(j) - y(j)$) if $\tilde{t}(j) = t(j)$ (resp. $1 - t(j)$).

Corollary 1. *y can be an integer solution of (1.1) only if $\tilde{y}(\pi) \neq 0$.*

Corollary 2. *If $\pi = \emptyset$, the condition is vacuous. If $d = 1$, one may reduce the bound interval of $y(\pi)$ by 1. If $d > 1$, one may reduce one of the d bound intervals of $y(\pi)$ by 1.*

When the reduction procedures of this section are preceded by those of Section 4.1, however, one assures that $d \neq 1$.

All techniques based on contraction, penalties, propagation, etc., clearly remain valid in some modified form.

(b) *Implementation.* In the experimentation for this paper we only implemented simple procedures based on contraction, as are described in what follows.

The entries of Q give information about the effect of an enumeration branch from a node ν to its successor node $\nu + 1$. Let

$$m1(j) = \#\{q(k,j),\ k : q(k,j) < 0\}$$
$$m2(j) = \#\{q(k,j),\ k : q(k,j) > 0\}. \tag{4.8}$$

It is clear that:

$m1(j) > 0$ implies $d^{\nu+1} < d^{\nu}$ if one branches with $y(j) = 0$,

$m2(j) > 0$ implies $d^{\nu+1} < d^{\nu}$ if one branches with $y(j) = 1$.

Such branches are called *contracting* branches, for they lead the search to a successor point in the integer lattice at which the problem is more constrained than before the branch.

The most favorable case is that of *double contraction*, which arises for branch j when:

$$m1(j) > 0 \quad \textit{and} \quad m2(j) > 0.$$

The enumerative code described here isolates a set, J, of candidates for branching according to the priorities:

(i) $J = \{j \mid m1(j) > 0$ and $m2(j) > 0\}$ ··· double contraction,

(ii) $J = \{j^* \mid m1(j^*)$ or $m2(j^*) \geq 1$ and equal to $\max_j \{m1(j), m2(j)\}\}$ (4.9)

(iii) $J = \{\text{all } j : y^*(j) \neq y^{\nu}(j)\}$ ··· "free" variables

In all cases, a branch with variable j is chosen such that $y^{\nu+1}(j) = 1$ (0) if $y^*(j) = 0$ (1).

As explained in [16], this requirement may necessitate a replacement of (4.5) (which represents "free" reduction with no state imposed) by a reduction after imposition of a state on (1.1), if the original preferred inequality system has no row for which all indicated branches lead away from the state.

Whether a procedure which ensures branching *away* from the state is indeed desirable, is not entirely clear. Some of our results in [10] seem to go against such a conjecture.

Our experimental system has been designed to admit the use of a truly "*dynamic state*", i.e. a state which can be recomputed (most likely so as to satisfy as closely as possible, in some sense, a set of minimal preferred inequalities) at *each* iteration. Such a feature, however, has not yet been tested.

5. Experiments

5.1 The experimental algorithm

The experimental algorithm was that of Section 2 with the techniques of Sections 3 and 4, stressed as follows.

one linear program was resolved and retained as per Sections 3.1 and 3.2;

the reduction procedures of Section 3.4 were used after any bound change, for whatever reason (possibly leading to some redundant work), in the order *a* and *b*;

states were determined as per Section 2.2 (see below for details);

a local search was conducted at every iteration (see Section 3.5), except for LEV = 0 (see below);

forward branches in that enumeration were chosen so as to lead to a successor with improved feasibility. Preferred inequalities were invoked for branching only if no improved feasibility was attained in the local search.

5.2 Experimental results

In Table 1 we describe a few test problems for which experimental results are described in Tables 2, 3, 4 and 5. The LP (Linear Programming) Objective Functions given are those obtained after an initial preprocessing reduction phase.

Table 1. Description of test problems

(m, n) Source	LP solutions in (0, 1)	LP solutions in (0, 2)	Integer solutions in (0, 1)	Integer solutions in (0, 1, 2)
(1) 6,12 [3]	6.65	6.58	13	11
(2) 10,20 [3]	−6155.3	−6623.5	−6120	−6570
(3) 28,35 [12]	521.05	358.23	550	400
(4) 12,44 [12]	56.68	56.68	75	—
(5) 5,39 [3]	10672	10737	10620	—
(6) 20,35[b]	31.34	27.71	47	

[a] Problem differs slightly from source problem.
[b] Seems to have originated as test problem in IBM Paris.

The problems are small. They are from either [3] or [12] (with some coefficients possibly altered by transcription errors), and are for the most part easily resolved as (0, 1) problems. However, we solved some of them also as (0, 1, 2) problems. Our experimental work was on an APL system time-shared with some 100 users. Hence even small problems require substantial on-line time, and in a sense our environment was not much different from that of a user with larger problems and greater computing power via a dedicated machine.

Table 2. Problem (6,15)

	BND	STA	LEV	NR	R1	Q1	Q2
(1)	1	LP, S	1	990 (155)	3 5	3 5	2 5
(2)	1	LP, S	2	72 (140)	2 3	2 3	2 3
(3)	2	LP, S	1	· · ·	15 21	17 [illegible]	—
(4)	2	LP, S	2	· · ·	—	4 11	—
(5)	1	ALWL	1	· · ·	4 [illegible]	4 [illegible]	4 [illegible]
(6)	1	ALWL	2	· · ·	[illegible]	4 [illegible]	
(7)	2	ALWL	1	· · ·	8 15	8 [illegible]	—
(8)	2	ALWL	2	· · ·	—	—	—
(9)	1	ALWU	1	· · ·	15 17	12 13	[illegible]
(10)	1	ALWU	2	· · ·	17 18	11 13	—
(11)	2	ALWU	1	· · ·	44 53	38 46	38 49
(12)	2	ALWU	2	· · ·	4 51	34 46	

— stands for "not run"
· · · stands for "not run, believed to be a bad strategy"

Table 3. Problem (10,20)

	BND	STA	LEV	NR	R1	Q1	Q2
(1)	1	LP, S	1	· · ·	57 57	51 51	1 11
(2)	1	LP, S	2	−6100 (225)	[illegible] 9	6 9	[illegible] 9
(3)	2	LP, I	1	· · ·	[illegible] 21	1 16	
(4)	2	LP, I	2	6570 (175)	[illegible] 15	1 23	1 17
(5)	1	ALWL	1	· · ·			
(6)	1	ALWL	2	· · ·	4158 (60)	−[illegible] (52)	· · ·
(7)	2	ALWL	1	· · ·	[illegible]	−3950 (140)	· · ·
(8)	2	ALWL	2	· · ·		· · ·	· · ·
(9)	1	ALWU	1	· · ·		· · ·	· · ·
(10)	1	ALWU	2	6120 (75)	[illegible]	[illegible]	3880 (65)

Table 4. Problem (20,45)

	BND	STA	LEV	NR	R1	Q1	Q2
(1)	[illegible]	LP, S	1	—	1 9	[illegible] 9	1 11
(2)	[illegible]	LP, S	2		1 15	2 [illegible]	1 11
(3)	2	LP, I	1	—	47 57	[illegible] 29	675 (35)
(4)	2	LP, I	[illegible]	—	[illegible] 25	5 25	
(5)	1	ALWL	[illegible]	· · ·	· · ·	1075 (50)	[illegible]
([illegible])	1	ALWU	1	—	—	[illegible] 45	—
(11)	2	ALWU	1		· · ·	1400 (50)	· · ·

Table 5. Diverse problems run with Q1

Type	Problem	BND	STA	LEV	Q1	
2	(12,41)	1	LP, .5	2	1	357
3	(5,29)	1	LP, .5	2	-10618	[60]
3	(20,28)	1	LP, .5	2	47	[800]

Tables 2, 3 and 4 are devoted to one sample problem each, run in a number of different ways. Table 5 gives a few results for somewhat more difficult problems. The (20, 28) problem, for example, requires on the order of one hundred LP programs, even when cutting plane techniques are used in branch-and-bound programming.

The columns headed BND and STA describe what bounds were used (on all variables) and what state was utilized. In all cases the state was computed at node 1 and updated in the obvious manner. The entry (LP, r) signifies that the state was obtained by rounding with the rounding parameter r. Column LEV refers to the search, which was used in practically all runs with a search level of 1 or 2. Search level 0 would correspond to no search.

The last columns are devoted to comparisons among four possible methods:

NR — no reduction method used at all.

R1 — reduction used as in Section 4.1.

Q1 — full reduction, Sections 4.1 and 4.2.

Q2 — full reduction; reduction (not search) used for strategy.

Under each of these 4 column headings there are two entries, NS, NT:

NS — iteration number ν at which optimal solution found;

NT — iteration number at which optimality ascertained.

However, when the second entry is in parentheses, this is meant to signify that the run did not terminate properly but was interrupted. In that case, (NT) is the iteration number at interruption and NS is the best objective function value found during the run (with some default value such as 9999, when no solution found).

The difference between Q1 and Q2 needs to be explained a little further. Having several features in an enumerative system often makes comparisons of results quite difficult. Accumulated counts of the successes of a particular technique depend strongly on the use of other techniques and on the order in which these techniques were deployed. In this particular instance, the fact that a search was used in all runs makes the subsequent use of minimal preferred inequalities somewhat ineffective. Column Q2 refers to runs in which the use of the search (to indicate indices of variables which lead to improved solutions or to points of reduced infeasibility) was suppressed.

It is not difficult to interpret the results, even though it is a little disappointing that there is so little difference among R1, Q1, Q2:

(i) The state has a very great influence on the enumeration, even if only used in a simple manner, as here. (Dynamic state determination is clearly of interest.) This is not new (compare [13]), but often forgotten. We believe this to be on account of the fact that standard branch and bound methods work in the neighborhood of LP solutions and are therefore often successful. This does not, however, negate the great importance of codes which permit the "*imposition*" of a state (not easily possible in BB programming) for full enumeration or for heuristic programming.

(ii) Using a local search can, but need not, make a big difference. Occasionally a 1-level search misses a good solution which can be found by a 2-level search, and then the search procedure can meander uselessly for quite a while before leading back to a lattice area of interest.

(iii) Search method NR, no reduction, was no good at all. To mitigate this result somewhat, we must consider, however, that NR was obtained by cutting out all reduction. Any decent enumerative code has some provision for making inferences about fixing variables or reducing bounds. NR is, therefore, not representative of a reasonable enumerative code, in spite of containing state and search features.

(iv) There appears to be only a slight improvement due to Q1 or Q2 over R1. Our conclusion is that using a local search feature for R1 renders somewhat ineffective the use of logical inequalities for strategic purposes. Other evidence makes it fairly clear that penalties ought to be invoked in conjunction with logical inequalities.

(v) It need hardly be emphasized that integer programming remains always unpredictable. Comparing row 1 and row 10 of Table 3, for example, one sees that in one case method Q2 is much better than Q1, in the other case much worse. This only confirms the well-known impossibility of finding *one* suitable algorithm for all problems.

Finally, one might be able to make a genuine case for the use of interactivity in integer programming. Watching the behaviour of the search, e.g. by printing out List 1 and List 2 of Section 3.1, plus some selected data on obj. function, bounds and infeasibilities, $(y(i) < 0)$, does give a feeling as to whether one is doing well or not.

For example, when one follows the behaviour of a search carefully, one will almost always notice that the "depth" of the *overall* search tree (namely the maximal i attained, before reduction procedures lead to backward steps) is a good indicator of whether things go well or not. Large values of i should perhaps be taken as poor behaviour and lead to redefinition of state, local search level or other parameters of the enumerative code. (In our past experience this behaviour was most evident with large plant location problems [13], where the use of a good state led otherwise quite difficult problems with 100 plants to have a search tree with level practically always below three. Given the relatively small machine at our disposal at that time, we did not resolve the problem but still believe that the search was "well behaved".)

References

[1] E. Balas and R. Jeroslow, Canonical cuts on the unit hypercube, SIAM J. Appl. Math. 23 (1972) 61–69.

[2] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Math. 4 (1962) 238–252.

[3] B. Bouvier and G. Messoumian, Programmes linéaires en variables bivalentes, Thesis, Faculté des Sciences, Univ. Grenoble, 1965.

[4] R. Breu and C.-A. Burdet, Branch and bound experiments in zero-one programming, Math. Programming Study 2 (1974) 1–50.

[5] R.E. Gomory and E.L. Johnson, Some continuous functions related to corner polyhedra, IBM Res. Report RC3311, Feb. 1971 (see also two recent papers in Math. Programming).

[6] F. Granot and P.L. Hammer, On the use of boolean functions in 0-1 programming, Methods of Operations Research, Vol. 12 (1972).

[7] M. Guignard, Preferred shadow prices in 0-1 programming, Res. Report, Dept. Stat. & OR, Wharton School, Univ. of PA, 1974.

[8] M. Guignard and K. Spielberg, Search techniques with adaptive features for certain integer and mixed-integer programming problems, Proc. IFIPS Congress, Edinburgh (North Holland, 1968).

[9] M. Guignard and K. Spielberg, The state enumeration method for mixed zero-one programming, IBM Phil. Sc. Center Rep. 320-3000, Feb. 1971.

[10] M. Guignard and K. Spielberg, A realization of the state enumeration procedure, IBM Phil. Sc. Center Rep. 320-3025, June 1973.

[11] E.L. Johnson and K. Spielberg, Inequalities in branch and bound programming, in: eds., R.W. Cottle and J. Krarup, Optimization Methods (English Univ. Press, London, 1974).

[12] C. Lemke and K. Spielberg, Direct search algorithms for zero-one and mixed-integer programming, J. ORSA, 15 (1967) 892–914.

[13] K. Spielberg, Plant location with generalized search origin, Management Sci. 16 (1969) 165–178.

[14] K. Spielberg, Minimal Preferred Variable Reduction for Zero One Programming, IBM Phil. Sc. Center Rep. 320-3013, July, 1972.

[15] K. Spielberg, A minimal inequality branch-bound method, IBM Phil. Sc. Center Rep. 320-3024, June 1973.

[16] K. Spielberg, Minimal preferred variable methods in zero-one programming, Working Paper, IBM Scientific Marketing, Dec. 1973.

[17] S. Zionts, Generalized implicit enumeration using bounds on variables for solving linear programs with zero-one variables, Naval Res. Logistics Quarterly 19 (1972) 165–181.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 287–297

SUBDEGREES AND CHROMATIC NUMBERS OF HYPERGRAPHS

Pierre HANSEN

Institut d'Economie Scientifique et de Gestion, Lille, France and Faculté Universitaire Catholique de Mons, Belgium

The chromatic number $\chi(H)$ of a hypergraph H is studied in relation with the degrees of the vertices of H and of the section hypergraphs of H. The *subdegree* of a vertex x_j of H is defined as the smallest integer k such that a sequential suppression of the vertices of degree $\leq k$ suppresses x_j. Bounds on the chromatic number $\chi(H)$ and on the independence number $\alpha(H)$ of H are obtained in terms of subdegrees. An algorithm for coloring H in $\chi(H)$ colors is proposed and computational experience is reported on.

1. Upper bounds on the chromatic number of a hypergraph

Chromatic numbers of graphs have been extensively studied both from the theoretical and from the computational points of view (see e.g. [1, 3, 17–19, 21]). Only some of the results obtained for graphs have been generalised to hypergraphs and very few algorithms have been proposed for determining the chromatic number of a hypergraph. Berge [1], Tomescu [22, 23] and Chvátal [4] have given upper bounds on the chromatic number of a hypergraph, as defined by Erdös and Hajnal [5]. Lovász [16] has shown that the theorem of Brooks holds for uniform hypergraphs, under some restrictions. Fournier and Las Vergnas [8, 9, 11], among others, have studied bichromatic hypergraphs. Extremal problems on uniform bichromatic or r-chromatic hypergraphs (k-graphs with property B or with property B_r) have been extensively studied. Results and references are given in chapter 4 of the book of Erdös and Spencer [6] "Probabilistic Methods in Combinatorics" and in a recent paper of Johnson [7].

Nieminen [20] has shown how the chromatic number of a hypergraph could be determined with a linear program in 0–1 variables; as both the numbers of variables and of constraints of that program are large the approach is more theoretical than practical. A heuristic algorithm for obtaining an approximation of the strong chromatic number of a hypergraph has been given by Laurière [12].

In part one of this paper the *subdegrees* of the vertices of a hypergraph are defined. This concept allows us to reformulate and extend to hypergraphs a result obtained for graphs independently by Matula [17] and by Szekeres and Wilf [21]. Then an upper bound on the chromatic number of a hypergraph due to Tomescu [22, 23] and a lower bound on the independence number of a hypergraph, due to

Lorea [14, 15] are strengthened. The proofs of these last results are different and much shorter than the original proofs. In part two an exact algorithm for determining the chromatic number of a hypergraph H is proposed and computational experience is reported on. Recall a *hypergraph* $H = (X, \varepsilon)$ ([1, 2]) is a couple where $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of *vertices* and $\varepsilon = \{E_1, E_2, \ldots, E_m\}$ is a finite family of non-empty subsets of X, the union of which is X, called *edges*. An edge E_j is *incident* to a vertex x_i if and only if $x_i \in E_j$. The *partial hypergraph* of $H = (X, \varepsilon)$ generated by a family $F \subset \varepsilon$ is the hypergraph $H_F = (X_F, F)$ where $X_F = \bigcup_{E_j \in F} E_j$.

The *section hypergraph* of $H = (X, \varepsilon)$ generated by a set $A \subset X$ is the partial hypergraph $H \times A = (A, \varepsilon_A)$ where $\varepsilon_A = \{E_j \mid E_j \in \varepsilon, E_j \subset A\}$.

A subset $S \subset X$ of vertices of H is *independent* if there is no E_j with $|E_j| > 1$ such that $E_j \subset S$; the *independence number* $\alpha(H)$ of H is the maximum cardinality of an independent set of H. A *coloration* of H is a partition of X into independent sets; the *chromatic number* $\chi(H)$ of H is the smallest number of independent sets in a coloration of H. The *degree* $d_H(x_i)$ of a vertex x_i of H is the maximum number of edges different from $\{x_i\}$ forming a partial family $\{E_k, k \in K\}$ with

$$E_k \cap E_l = \{x_i\} \qquad (k, l \in K; k \neq l).$$

Let us call *suppression* of a vertex x_i of H and of all edges incident to x_i the replacement of H by its section hypergraph generated by $X - x_i$. Let us define the *subdegree* $d'_H(x_i)$ of a vertex x_i of H as the smallest integer k such that a sequential suppression of all vertices of degree $\leq k$ and of all edges incident to those vertices in H (or in the section hypergraphs of H obtained after the first suppression) suppresses x_i. Clearly $d'_H(x_i) \leq d_H(x_i)$ for all i; hence the name subdegree. The subdegrees of a hypergraph can be computed by suppressing a vertex of minimum degree and the incident edges in H, then a vertex of minimum degree in the resulting hypergraph and so on.

Note that the order of suppression of the vertices may not be unique but that the values of the subdegrees are unaffected by this order. Let us call *subdegree order* the reverse order of the order of suppression of the vertices of H when the subdegrees are computed.

Theorem 1. *Let h' denote the maximum subdegree of a hypergraph $H = (X, \varepsilon)$; then*

$$\chi(H) \leq 1 + h' = 1 + \max_{A \subseteq X} \min_{x_i \in A} d_{H \times A}(x_i) \tag{1}$$

where $H \times A = (A, \varepsilon_A)$ is the section hypergraph of H generated by $A \subset X$ and $d_{H \times A}(x_i)$ denotes the degree of x_i in $H \times A$.

Proof. Consider a sequential coloration of the vertices of H in a subdegree order, each vertex being assigned the first color such that no edge has all of its vertices of the same color. Assume the vertices $x_1, x_2, \ldots, x_{i-1}$ are colored; as the first uncolored vertex, x_i, is incident to at most h' edges colored in one color (except at x_i) it can always be colored in one of $1+h'$ colors; by iteration the inequality part of (1) is obtained. To prove the equality part note the right-hand side of (1) cannot be lower than $1+h'$ as there exists at least one section hypergraph $H \times B = (B, \varepsilon_B)$ such that $\min_{x_i \in B} d_{H\times B}(x_i) = h'$ by definition of h'. Assume there exists a section hypergraph $H \times C = (C, \varepsilon_C)$ such that $\min_{x_i \in C} d_{H \times C}(x_i) > h'$.

Let x_k be the last vertex in the subdegree order to belong to C, let D denote the set consisting of x_k and all preceding vertices in the subdegree order and let $H \times D = (D, \varepsilon_D)$ denote the section hypergraph generated by D. As $C \subset D$, $\varepsilon_C \subset \varepsilon_D$ and $d_{H\times C}(x_k) \le d_{H\times D}(x_k) \le h'$, a contradiction.

Theorem 2. *Let $(S_1, S_2, \ldots, S_q)$ denote a partition of the set X of vertices of $H = (X, \varepsilon)$ in q independent sets and let*

$$d'_k = \max_{x_i \in S_k} d'_H(x_i) \qquad (k = 1, 2, \ldots, q). \tag{2}$$

Then

$$\chi(H) \le \max_{k \le q} \min\{k, d'_k + 1\}. \tag{3}$$

Proof. Let k^* denote the value of the right-hand side of (3) and $S'_k = S_k \cap \{x_i : d'_H(x_i) \ge k^*\}$, $k = 1, 2, \ldots, k^*$. By coloring the vertices of $S'_1, S'_2, \ldots, S'_{k^*}$ in the colors $1, 2, \ldots, k^*$ respectively, all vertices such that $d'_H(x_i) \ge k^*$ are colored. The remaining vertices can then be colored sequentially in the subdegree order without introducing any new color.

The subdegrees of the vertices of a hypergraph may be much smaller than their degrees. For instance, it is easily shown, by a similar argument as in the proof of the lemma of [10], that the subdegrees of all the vertices of a hypergraph without cycles of length greater than two are equal to one; the degrees of the vertices of such a hypergraph may be arbitrarily large.

Corollary 2.1. *Let h' denote the maximum subdegree of a hypergraph $H = (X, \varepsilon)$ and $n = |X|$; then*

$$\alpha(H) \ge \left[\frac{n}{h'+1}\right]^* \tag{4}$$

(where $[a]^*$ denotes the smallest integer greater than or equal to a).

Proof. As H can be colored in $\chi(H) \le 1 + h'$ colors, X can be partitioned in $\chi(H)$ independent sets and at least one of these sets contains at least $[n/(h'+1)]^*$ vertices.

2. An algorithm for the chromatic number of a hypergraph

The constructive proofs of Theorems 1 and 2 may be viewed as heuristic algorithms for coloring a hypergraph H in $\chi(H)$ or slightly more colors; these heuristic algorithms could be combined with branch-and-bound in order to obtain exact algorithms for the coloration of H in $\chi(H)$ colors. Such a procedure would probably be computationally inefficient as the determination of the subdegrees $d_H(x_j)$ of the vertices of H involves the resolution of a large number of *packing problems*. We therefore propose the following simple branch-and-bound algorithm:

(a) *Initialisation*. Set n_{opt}, number of colors in the best known solution, equal to $|X|+1$. Consider all vertices of H as uncolored, with no forbidden colors, and no edges of H as eliminated.

(b) *Resolution test*. If all edges have at least two vertices of different color, and have been eliminated, note the current coloration in C_{opt}, update n_{opt} and go to (g).

(c) *Direct optimality test*. If for an uncolored vertex the number of forbidden colors is equal to $n_{opt}-1$, go to (g).

(d) *Conditional optimality test*. If for an uncolored vertex x_j the number of forbidden colors is equal to $n_{opt}-2$, go to (f).

(e) *Selection of a vertex to be colored*. Select the uncolored vertex x_j for which the most colors are forbidden; in case of ties select among the tied vertices that one which belongs to the most non-eliminated edges, weighted by the inverse of their number of uncolored vertices.

(f) *Coloration of a vertex*. Seek the first color q not forbidden to x_j; note if this color has already been used or not; assign color q to x_j. Consider all non-eliminated edges containing x_j: if an edge E_l has a vertex x_k colored in a different color than q, eliminate it; if an edge E_l has all its vertices but one colored in color q, seek the uncolored vertex x_s belonging to E_l and forbid color q to x_s (if it has not yet been done). Then go to (b).

(g) *Backtracking*. If coming from (b) uncolor the vertices in the reverse order of their coloration until the last color used disappears. Uncolor the last vertex chosen at step (e) and forbid the color used to that vertex; uncolor all vertices colored after that one.

Update the tables of forbidden colors and of eliminated edges. If at least one vertex remains colored, go to (b). Otherwise, an optimal coloration C_{opt} of H in n_{opt} colors has been found (any uncolored vertex may be assigned any of the colors used).

The algorithm described above has been programmed in Fortran Extended and tested on a CDC 6500 computer. All information (i.e. forbidden colors, eliminated edges, etc.) is updated from iteration to iteration and not recomputed. 60 test problems have been solved; the hypergraphs have 20 to 40 vertices and 200 randomly generated edges; in the series 1 (respectively 3) 100 edges have 2 vertices (resp. 3 vertices) and 100 edges have 3 vertices (resp. 4 vertices); in the series 2 and 4 all edges have 3 and 4 vertices respectively. The results of these experiments are summarized in Table 1. The algorithm appears to be efficient for coloring small hypergraphs.

Table 1.

Problem series	N_V	N_C	N_{ST}	N_B	T
1	20	5.0	31.6	2.4	0.764
	30	4.2	72.6	6.4	1.781
	40	4.0	47.0	1.6	1.679
2	20	3.2	178.0	28.6	3.448
	30	3.0	78.8	6.8	2.095
	40	3.0	74.0	5.2	2.575
3	20	3.0	41.4	4.6	1.024
	30	3.0	75.8	7.0	2.130
	40	3.0	125.8	9.6	3.806
4	20	3.0	126.0	18.8	2.854
	30	3.0	711.2	91.6	15.780
6	40	2.0	1529.6	137.2	33.909

N_V = number of vertices of H, N_C = number of colors in C_{opt}, N_{ST} = number of vertices in the solution tree, N_B = number of backtracks, T = computation time in seconds CPU on CDC 6600, input and output times excluded; N_C, N_{ST}, N_B, T are averages for 5 problems.

References

[1] C. Berge, Graphes et hypergraphes (Dunod, Paris, 1970).

[2] C. Berge and D.K. Ray-Chaudhuri, eds., Hypergraph Seminar (Springer, Berlin, 1974).

[3] J.R. Brown, Chromatic scheduling and the chromatic number problem, Management Sci., 19 (1972) 456–463.

[4] V. Chvátal, Hypergraphs and Ramseyian theorems, Proc. Am. Math. Soc., 27 (1971) 434–440.

[5] P. Erdős and A. Hajnal, On chromatic numbers of graphs and set systems, Acta Math. Acad. Sc. Hungaricae, 17 (1966) 61–99.

[6] P. Erdős and J. Spencer, Probabilistic Methods in Combinatorics (Academic Press, New York, 1974).

[7] D. Johnson, On property B, J. Combinatorial Theory, 20 (1976) 64–66.

[8] J.-C. Fournier and M. Las Vergnas, Une classe d'hypergraphes bichromatiques, Discrete Math., 2 (1972) 407–410.

[9] J.-C. Fournier and M. Las Vergnas, Une classe d'hypergraphes bichromatiques, II, Discrete Math., 7 (1974) 99–106.

[10] P. Hansen and M. Las Vergnas, On a property of hypergraphs without cycles of length greater than two, 99–101 in: [2].

[11] M. Las Vergnas, Sur les hypergraphes bichromatiques, 102–110 in [2].

[12] H. Lemière, Problèmes d'emploi du temps et algorithme de coloration des hypergraphes, Comptes Rendus Acad. Sci. Paris, 278 (1974) 1159–1162.

[13] D.R. Lick and A.T. White, k-degenerate graphs, Canadian J. of Math., 22 (1970) 1082–1096.

[14] M. Lorea, Bases des stables dans les hypergraphes, Comptes Rendus Acad. Sci. Paris, 275 (1972) 163–165.

[15] M. Lorea, Ensembles stables dans les hypergraphes, Cahiers du Centre d'Études de Rech. Opér., 14 (1972) 125–133.

[16] L. Lovász, On chromatic number of finite set systems, Acta Mathematica Acad. Sci. Hungaricae 19 (1968) 59–67.

[17] D.W. Matula, A min-max theorem for graphs with application to graph coloring, SIAM Review 10 (1968) 481–482.

[18] D.W. Matula, K-components, clusters and slicings in graphs, SIAM J. on Applied Math. 22 (1972) 459–480.

[19] D.W. Matula, G. Marble and J.D. Isaacson, Graph coloring algorithms, in: R.C. Read, ed., Graph Theory and Computing (Academic Press, New York, 1972) 109–122.

[20] J. Nieminen, A viewpoint to the minimum coloring problem of hypergraphs, Kybernetika 10 (1974) 504–508.

[21] G. Szekeres and H.S. Wilf, An inequality for the chromatic number of a graph, J. Combinatorial Theory 4 (1968) 1–3.

[22] I. Tomescu, Sur le problème du coloriage des graphes généralisés, Comptes Rendus Acad. Sci. Paris 267 (1968) 250–252.

[23] I. Tomescu, Méthodes combinatoires dans la théorie des automates finis, in: Gr. C. Moisil, ed., Logique, Automatique, Informatique (Editions de l'Académie de la République Socialiste de Roumanie), (1971) 267–424.

[24] K. Zarankiewicz, Sur les relations symétriques dans l'ensemble fini, Colloquium Math. 1 (1947) 10–15.

Annals of Discrete Mathematics 1 (1977) 293-330
© North-Holland Publishing Company

CUTTING-PLANE THEORY: DISJUNCTIVE METHODS

R.G. JEROSLOW
GSIA and Department of Mathematics, Carnegie-Mellon University, Pittsburgh PA 15213, U.S.A.

This paper is a survey, with new results, of the disjunctive methods of cutting-plane theory which were devised by Balas, Glover, Owen, Young, and other researchers, over the past half decade. The basic disjunctive cut principle is derived, its interrelations with the other cut-producing procedures are discussed, and applications of it are given. Many theorems from the literature are concisely proven, and a fairly complete bibliography is provided. In addition, several new results are presented, and finitely convergent disjunctive cutting-plane algorithms are given for a wide class of programs.

0. Introduction

This paper is a survey, with new results, of the disjunctive methods in cutting-plane theory that have been devised [1, 4, 15, 21, 24, 25, 32, 43, 50] and developed by several authors (e.g., [2, 5, 9, 10, 11, 12, 22, 26, 34, 35, 37, 51]).

The new results presented here include: a broad sufficient condition for the disjunctive cuts to provide all the valid cutting planes (Section 1.1.1, joint with C.E. Blair); a sufficient condition for distributivity in the co-propositions that are used to express the general cut-form of disjunctive cuts (Section 2.1.2); and proofs of finite convergence for a class of cutting-plane algorithms that use disjunctive cuts (Section 2.2). As we will show in a later paper, the distributivity result of Section 2.1.2 provides a finitely-convergent cutting-plane algorithm for a class of problems that includes the linear complementarity problem.

In terms of expository presentation, we begin in Section 1.1 with the basic principle of disjunctive constraints, as presented in the format of [3, 4]. We then relate this principle to the earlier one of the intersection/convexity cuts and show that it is stronger (Section 1.2). We then discuss the connection between this principle and the "polyhedral annexation" method of [24]. Following this, the Lagrangean relaxations for integer programs are interpreted from the point of view of disjunctive cuts (Section 1.3), and in turn disjunctive cuts are interpreted as cuts obtained from sublinear and subadditive functions (Section 1.4). Next, three examples are given illustrating various uses of disjunctive cuts (Section 1.5). Finally, we give a compact presentation of the "co-propositions" of [37], which represent a systematic development of the disjunctive cut principle, and we mention general properties of the co-propositions, some for the first time (Section 2.1).

References to the literature are given in the discussions of the appropriate subsections following the statement of results.

The background necessary for this paper is a knowledge of the Duality Theorem of Linear Programming, and polarity for polyhedra. Either [45] or [47] are good general references, though for polarity [49] still deserves reading (see also [13]). For Section 1.4 only, we assume some acquaintance with [36]. The paper is intended to be read consecutively.

This paper is a revised version of Part II of [35]. A companion paper [36], which is Part I of [35], surveys the algebraic methods of cutting-plane theory, and gives new results. Some proofs are omitted; these are usually supplied in [35].

1. The basic disjunctive cut principle and its relationship to other principles and approaches

The disjunctive methods consist of various ways that one can obtain cutting-planes from logical constraints on linear inequalities. They use, in various forms, a certain basic principle. This principle is equivalent, in some contexts, to a cutting-plane formulation of certain enumerations or partial enumerations.

In what follows, the pointwise supremum $\sup_{h \in H} v^h$ of a set of vectors $\{v^h \mid h \in H\}$, $v^h = (v_1^h, \ldots, v_r^h)$ for $h \in H$, denotes that vector $v = (v_1, \ldots, v_r)$ such that

$$v_j = \sup_{h \in H} v_j^h, \quad j = 1, \ldots, r. \tag{1.A}$$

The writing of an expression $\sup v^h$ entails that each supremum in (1.A) is finite.

1.1. The basic disjunctive cut principle

Theorem. *Suppose that at least one of the linear inequality systems*

$$\begin{aligned} &A^h x \geq b^h, \\ &\quad x \geq 0, \qquad (h \in H) \end{aligned} \tag{S_h}$$

must hold. Then for any choice of non-negative vectors $\lambda^h \geq 0$ the inequality

$$\left(\sup_{h \in H} \lambda^h A^h\right) x \geq \inf_{h \in H} \lambda^h b^h \tag{DC}$$

is valid. Furthermore, if every system (S_h) is consistent, then for any valid inequality

$$\sum_{j=1}^{n} \pi_j x_j \geq \pi_0 \tag{1.1.A}$$

there are non-negative vectors $\lambda^h \geq 0$, $h \in H$, such that $\pi_0 \leq \inf \lambda^h b^h$ and, for

$j = 1, \ldots, r$, the jth component of $\sup \lambda^h A^h$ *does not exceed* π_j. *In this principle, H may be infinite or finite.*

Proof. Toward the forward direction of the principle, note that, since at least one system (S_h) holds for any x, at least one inequality $(\lambda^h A^h)x \geq \lambda^h b^h$ holds; but h may depend on x. Taking the supremum and infimum in (DC) removes this dependency, and is still valid since $x \geq 0$.

Toward the converse, assume that all systems (S_h) are consistent. Since (1.1.A) is implied by any one of them, by the duality theorem for any $h \in H$ there exist $\lambda^h \geq 0$ satisfying $\lambda^h b^h \geq \pi_0$ and $(\lambda^h A^h)_j \leq \pi_j$, $j = 1, \ldots, r$. Here, $(\lambda^h A^h)_j$ is the jth component of $\lambda^h A^h$. Taking infima, we obtain $\pi_0 \leq \inf \lambda^h b^h$. Taking suprema, we obtain $(\sup \lambda^h A^h)_j \leq \pi_j$, $j = 1, \ldots, r$. Q.E.D.

In the applications, one deduces the systems (S_h), at least one of which must hold, from the constraints of the integer program

$$\begin{aligned} &\inf \ cx \\ \text{subject to } \ &Ax = b, \\ &x \geq 0, \\ &x \text{ integer} \end{aligned} \tag{IP}$$

A trivial application is to use as (S_h) the constraints

$$\begin{aligned} Ax &= b \\ x &= h \\ x &\geq 0 \end{aligned} \tag{S_h)'}$$

where $h = (h_1, \ldots, h_r)$ is a non-negative integer vector, and all such vectors (or, at least all feasible ones, assuming (IP) consistent) are enumerated as $h \in H$ varies. In principle, then, all valid cuts for (IP) become available, as one uses disjunctive systems (DC) expanded further and further toward $(S_h)'$. As we shall see below, there is more here than simply a similarity with branch-and-bound in the context of $(S_h)'$.

Note that, if some inequalities of (S_h) are replaced by equalities, the corresponding multiplier is unrestricted in sign.

The forward direction of the above principle, for the special case when A^h has a single row and H is finite, was stated by Owen [43]. Balas [3, 4] stated the forward direction of the general principle; see [37] for the reverse direction. Balas [5, 6] also extended the principle to the case that $x \geq 0$ is not required to occur among the constraints of (S_h). Another generalization is in [37], and related results are in [23, 24, 25, 34] (more on Glover's alternate format for disjunctive cuts in Section 1.2.1 below).

1.1.1. A result on the converse to the disjunctive cut principle

A converse to the disjunctive cut principle is a statement that, under suitable hypotheses, the principle gives all the valid cuts.

Not all the systems (S_h) need be consistent for the disjunctive cut principle to give all valid cuts; in [9] a general result is given for the converse, which we repeat here. For a different converse, see [5, Theorems 4.2 and 4.7].

For $h \in H$, define the recession cone C_h by

$$C_h = \{x \mid A^h x \geq 0, x \geq 0\}. \tag{1.1.1.A}$$

Theorem. *The disjunctive cut principle (DC) gives all valid cuts for the logical condition that at least one (S_h) holds, if for every $h \in H$ such that (S_h) is inconsistent, we have*

$$C_h \subseteq \sum \{C_p \mid p \in H \text{ and } (S_p) \text{ consistent}\}, \tag{1.1.1.B}$$

(with the summation interpreted as $\{0\}$ if all (S_p) are inconsistent).

Proof. To prove the stated result, it clearly suffices, for (S_h) inconsistent, to find $\lambda^h \geq 0$ with $\lambda^h A^h \geq \pi$ $(\pi = (\pi_1, \ldots, \pi_n))$ and $\lambda^h b^h \geq \pi_0$, for any cut (1.1.A) valid for all the consistent systems (S_p); then the taking of maxima and a minimum as in the proof of 1.1 completes the proof.

Note that, if (S_p) is consistent and (1.1.A) is valid, we have $\pi x \geq 0$ for $x \in C_p$. This follows from the fact that $\lambda^p A^p \geq \pi$ for some $\lambda^p \geq 0$, and the fact that $A^p x \geq 0$. But then if $x \in C_h$ with (S_h) inconsistent, writing

$$x = \sum \left\{ x^{(p)} \,\middle|\, \begin{array}{l} p \in H, (S_p) \text{ consistent} \\ \text{and } x^{(p)} \in C_p \end{array} \right\}$$

for certain $x^{(p)} \in C_p$ by (1.1.1.B), we have

$$\pi x = \sum \pi x^{(p)} \geq 0. \tag{1.1.1.C}$$

Also, (1.1.1.C) is trivial if all (S_p) are inconsistent.

Therefore, $\pi x \geq 0$ is implied by $A^h x \geq 0$, $x \geq 0$, and by the Farkas Lemma, we obtain the multipliers $\theta^h \geq 0$ with $\theta^h A^h \geq \pi$. Finally, by the inconsistency of (S_h) there is $\rho^h \geq 0$ with $\rho^h A^h \leq 0$, $\rho^h b^h > 0$. But then for $r \geq 0$ suitably large, putting $\lambda^h = \theta^h + r\rho^h$, we have $\lambda^h A^h \geq \pi + 0 = \pi$, $\lambda^h b^h \geq \pi_0$ as desired. This completes the proof. Q.E.D.

As one application of this strengthened converse, if all the matrices A^h are identical and at least one system (S_p) is consistent, the converse will hold. For instance, in $(S_h)'$ there is no need to delete inconsistent systems.

For a second application, if at least one (S_h) is consistent and if, for all $h \in H$ there is some d^h for which $A^h x \geq d^h$, $x \geq 0$ is bounded and consistent, the converse holds. Note that the consistency and boundedness of $A^h x \geq d^h$, $x \geq 0$ implies that all $G_h = \{0\}$.

1.1.2. *Geometry of the disjunctive cut principle*

It is easy to see that the cuts (1.1.A) which hold if any systems (S_h) holds, are precisely those valid for the closed convex span clconv(T) of the set

$$T = \{x \geq 0 \mid \text{for at least one } h \in H,\ A^h x \geq b^h\}. \tag{1.1.2.A}$$

Indeed, (1.1.A) is valid on a closed convex set, hence valid for clconv(T) at least. But for any point $x \notin$ clconv(T), there is a separating hyperplane (1.1.A) valid for clconv$(T) \supseteq T$, which cuts off x.

1.1.3. *Earlier work on logical constraints for linear inequalities*

Since Dantzig's work in the early 1950's, it has been widely known that a primary use of integer variables in linear programs is to express logical restrictions that are placed on linear inequalities.

The disjunctive methods appear to be the first explicit use of this perspective toward constructing cutting planes. However, a result concerning linear inequalities constrained by logical requirements appears even earlier in [15, Appendix A], and we repeat it here.

Theorem [15]. *Suppose that every solution to*

$$Ax \geq 0$$

satisfies at least one of the homogeneous inequalities in the inequality system

$$Bx \geq 0.$$

Then there are vectors of multipliers $v \geq 0$, $z \geq 0$, with $z \neq 0$, for which we have

$$vA = zB$$

Note that if B has only one row, the theorem is the ordinary Farkas Lemma.

Since the publication of [15], Duffin has generalized the above theorem to treat inhomogeneous inequalities (private communication).

1.2. *The disjunctive cut principle and the earlier intersection/convexity cut principle*

Just as the algebraic approach has been called the "subadditive" approach, due to the recent emphasis on subadditivity as opposed to purely algebraic features, the disjunctive approach has other synonyms: convexity, intersection, geometric.

The new principle has evolved as a strengthening and generalization of an earlier principle, from which the other synonyms derive.

Theorem: *Let the convex body*

$$C = \{x \mid a^h x \leq b_h, h \in H\} \tag{1.2.A}$$

be defined by certain hyperplanes $a^h x \leq b_h$, $h \in H$, $a^h = (a_1^h, \ldots, a_r^h)$, and let S be an arbitrary subset of $\mathbb{R}^r$.

Suppose that each $b_h > 0$, and that

$$\bar{\lambda}_j = \sup\{\lambda \mid a^h \cdot (e_j \lambda) \leq b_h, h \in H\} \tag{1.2.B}$$

is non-zero for $j = 1, \ldots, r$, where e_j is the jth unit vector. Then if $S \cap C = \emptyset$, every point $x \in S$ with $x \geq 0$ satisfies the cut

$$\sum_{j=1}^{r} (1/\bar{\lambda}_j) x_j \geq 1, \tag{IC}$$

where $1/\infty = 0$.

Proof. Omitted (but see e.g., [1] for one proof, and below for a justification via the validity of (DC) in Section 1.1). Q.E.D.

The use of the intersection/convexity cut (IC) occurs where x of (1.2.A) are the non basic variables of the Simplex Tableau, and the co-ordinate system has also been translated so that the current linear programming vertex is at $x = 0$. S is taken to be the integer points of the structural space. Then (1.2.B) represents the intercept of the jth tableau edge with the boundary of C. The hypotheses then state that no integer point is in the interior of C. The cut (IC) is then the hyperplane passing through the intersection points of the extended tableau edges with the boundary of C, and it is oriented to "cut off" the current vertex $x = 0$.

The strength of an intersection cut depends on the shape and size of the convex set C. Based on this approach, several procedures were proposed for generating cuts from suitably chosen convex sets [2, 22, 25]. We will not review these cuts here, but will mention briefly that the outer polar cut of [2] was the first cutting plane in the literature to incorporate information from the problem constraints that are slack at the point of the feasible set from which the cut is generated.

The new principle of disjunctive cuts is a direct improvement upon the earlier one. To see the connection, note that the hypothesis of the theorem implies that at least one of the systems

$$\begin{aligned} a^h x &\geq b_h, \\ x &\geq 0, \end{aligned} \tag{S_h}$$

holds for $x \in S$ (as $S \cap C = \emptyset$). Therefore, setting $\lambda_h = 1/b_h$ in the disjunctive principle, we obtain as (DC) the cutting plane

$$\left(\sup_h a^h/b_h\right) x \geq 1. \tag{DC}$$

From (1.2.B),

$$\bar{\lambda}_j = \begin{cases} \inf\{b_h/a_j^h \mid a_j^h > 0\} \\ +\infty, \quad \text{if all } a_j^h \leq 0, \end{cases} \tag{1.2.C}$$

and so the jth coefficient of (IC) is

$$1/\bar{\lambda}_j = \begin{cases} \sup_h \{a_j^h/b_h \mid a_j^h > 0\}, & \text{if at least one } a_j^h > 0; \\ 0, & \text{if all } a_j^h \leq 0. \end{cases} \tag{1.2.E}$$

Therefore, this coefficient is the same as $\sup a_j^h/b_h$ from (DC)' whenever $a_j^h > 0$ for at least one $h \in H$. This is the case for all coefficients, whenever the convex set C is bounded. If, however, $a_j^h \leq 0$ for all $h \in H$, i.e., if C is unbounded and contains a whole edge, the jth coefficient of the earlier cut is 0, whereas that of the new cut may be negative. For a detailed discussion of the connections between the two principles see [4].

The real advantage of the new principle, however, lies less in the fact that it is theoretically stronger than the old one, than in the fact that it has proven easier to use. Much of the ingenuity, needed to devise situations where the principle applies, has been greatly reduced.

1.2.1. *Glover's format for disjunctive cuts*

An alternate procedure for obtaining disjunctive cuts is Glover's polyhedral annexation technique [24, 25].

This technique is to be repeatedly applied to a family of polyhedra $P_1, \ldots, P_k$. It is assumed that no point of a set S is in the interior of any P_k. The P_k typically represent the integrality constraints (e.g., P_k is a translate of $\{x \mid 0 \leq x_1 \leq 1\}$) or the reverse of an inequality constraint defining the feasible region). A single application of polyhedral annexation involves only two of $P_1, \ldots, P_k$ (plus additional polyhedra added to the list) which are selected for the application.

A single application is as follows. Assume that polyhedra Q and U are chosen with

$$Q = \left\{x \mid \sum_{j=1}^{r} a_{ij}x_j \leq a_{i0}, i = 1, \ldots, q\right\} \tag{1.2.1.A}$$

$$U = \left\{x \mid \sum_{j=1}^{r} b_{kj}x_j \leq b_{k0}, k = 1, \ldots, u\right\}. \tag{1.2.1.B}$$

Then one selects an annexation index i^* in $\{1, \ldots, q\}$ and one adds the polyhedron

$$W=\Big\{x \,\Big|\, \sum_{j=1}^{n} a_{ij}x_j \geq a_{i0},\ i\neq i^*;\qquad \sum_{j} (\theta_t a_{i^*j}+\lambda_t b_{tj})x_j \geq \theta_t a_{i^*0}+\lambda_t b_{t0}\Big\} \tag{1.2.1.C}$$

to the list of polyhedra, in addition to $P_1, \ldots, P_u$ and those previously added. In (1.2.1.C), the parameters $\theta_t, \lambda_t \geq 0$ may be arbitrarily chosen.

The interpretation of this single polyhedral annexation is as follows. Given inductively that Q and U have no points of S in their interior[1], and all $\lambda_t + \theta_t > 0$, then neither does W. In particular, one may use W to obtain a valid cut, either by the earlier intersection cut principle, or by its strengthened version.

Glover showed [25] that, in finitely many applications of polyhedral annexation, followed by the taking of an implied cut, any valid cut (1.1.A) for a bounded integer programming problem can be obtained. This result was announced in September 1973.

His proof reveals more, since a pivotal step is the following assertion. Putting

$$P_k=\Big\{x \,\Big|\, \sum_{j=1}^{n} a^k_{ij}x_j \geq b^k_i,\ i=1,\ldots,p_k\Big\} \tag{1.2.1.D}$$

in finitely many steps one obtains a polyhedron of the form

$$P=\Big\{x \,\Big|\, \sum_{j=1}^{n}\Big(\sum_{k=1}^{u} (\theta^\sigma_k a^k_{\sigma(k),j})x_j\Big) \geq \sum_{k=1}^{u} \theta^\sigma_k b^k_{\sigma(k)},\ \text{all } \sigma \in \Sigma\Big\} \tag{1.2.1.E}$$

for any multipliers $\theta^\sigma_k \geq 0$, where σ denotes a function with domain $\{1,\ldots,u\}$ such that $\sigma(k)$ is in $\{1,\ldots,p_k\}$ for $k=1,\ldots,u$, and Σ denotes the set of all such functions σ. I.e., $\sigma(k)$ picks out one of the constraints of P_k, so as σ varies over Σ, the $|\Sigma|$ constraints of P represent all possible different ways of making selections of constraints, one from each P_k.

The proof of this assertion is by induction on u, all $u \geq 2$ reducing to the case $u=2$. This case $u=2$ can also be done by induction on the number of constraints in the second polyhedron, say U of the pair Q, U above. The details are in [25].

Regarding this assertion, we note the following. The cut for P of (1.2.1.E), used by Glover to obtain cuts from the polyhedra developed in polyhedral annexation, is the disjunctive cut obtained from the assertion that at least one of the defining constraints of P goes the other way, i.e., that at least one of the inequalities

$$\sum_{j=1}^{n}\Big(\sum_{k=1}^{u} \theta^\sigma_k a^k_{\sigma(k),j}\Big)x_j \leq \sum_{k=1}^{u} \theta^\sigma_k b^k_{\sigma(k)},\quad \sigma\in\Sigma,\ x\geq 0, \tag{1.2.1.F}$$

holds. Indeed, no feasible point is in the interior of P.

However, the disjunctive cuts (DC) from (1.2.1.F), as the multipliers $\theta^\sigma_k \geq 0$ vary, are those obtained from the assertion that at least one of the systems

$$\sum_{j} a^k_{\sigma(k),j}x_j \leq b^k_{\sigma(k)},\quad k=1,\ldots,u, \tag{1.2.1.G$_\sigma$}$$

[1] Here a point is "interior" to an inequality system if it satisfies each inequality strictly.

holds. But if no feasible point is in the interior of any P_σ of (1.2.1.D), then indeed at least one of the systems (1.2.1.G)$_h$ holds — let $\sigma(k)$ denote the constraint of P_k violated by a feasible x for (IP), $k = 1, \ldots, n$.

In summary, the cuts showing the finiteness of the polyhedral annexation procedure, are the disjunctive constraint cuts that one obtains by converting the information regarding the P_σ (i.e., that they have no interior feasible points) into the form (S_h). Note that this conversion involves manipulating strings which may have exponential length, and hence is not a practical way of obtaining all cuts (DC) if $|H|$ is small.

The reverse reduction is also true. Specifically, given that at least one system (S_h) holds, letting σ denote a function which chooses one constraint from each system (S_h), and writing $A^\sigma = (a^h_\sigma)$, $b^\sigma = (b^h_\sigma)$, then there is no feasible integer point for (IP) in the interior of P_σ given by

$$P_\sigma = \left\{ x \geq 0 \;\middle|\; \sum_{j=1}^{n} a^h_{\sigma(h)j} x_j \leq b_{\sigma(h)h}, \; h \in H \right\}. \tag{1.2.2.H}$$

Indeed, for some $p \in H$ (S_p) holds, so in P_σ the constraint for $h = p$ is satisfied only in the direction $(\geq)$, while $(<)$ is needed for interior points. When $|H|$ is finite, the polyhedra P_σ for all σ can form the basis for a polyhedral annexation process which produces all cuts (DC), under suitable hypotheses analogous to those for the converse direction of the principle of 1.1 above. Note also that this reverse reduction also may involve string manipulations of exponential length, and hence is not a practical way of obtaining a polyhedral annexation cut that requires only a few annexation steps.

We have seen that either Balas' principle of 1.1, for the format (S_h), or Glover's principle for the format of polyhedral annexation, yield the same family of valid cuts when $|H|$ is finite in 1.1. Each is advantageous when information is presented in (or easily converted to) its format.

1.3. Lagrangean relaxations interpreted via disjunctive cuts

There different ways of using the principle of 1.1, and in important instances they are of different mathematical strength, with relative dominances ascertainable. In providing one example here, which motivates our constructions in 2. below, we will also obtain a new perspective on the Lagrangean relaxations, a particularly interesting discussion of these is in Geoffrion's paper [19], which also references work in that topic.

Given a set of consistent constraints in integer variables $x \geq 0$, it may be advantageous to partition these into two sets, the second of which has some special structure that one may be able to exploit:

$$\begin{aligned} Dx &\geq d \\ Ex &\geq e \end{aligned} \tag{1.3.A}$$

(This is the point of view of [19].)

Now we know that all valid cutting planes are obtainable from the disjunctive systems

$$Dx \geq d, \quad Ex \geq e, \quad x = h \tag{1.3.B$_h$}$$

as h varies over all non-negative integer vectors h, provided only that the optimization problem

$$\begin{aligned} &\min \quad cx \\ &\text{subject to} \quad Dx \geq d,\ Ex \geq e, \\ &\qquad\qquad\quad x \geq 0, \quad \text{and integer} \end{aligned} \tag{P}$$

is consistent. One may try therefore to obtain the best possible cut (i.e., to maximize π_0 in (1.1.A)) from the systems (1.3.B)$_h$ with $\pi \leq c$.

But this may be difficult, while in contrast it may be easier to optimize disjunctive cuts for the systems

$$Ex \geq e, \quad x = h \tag{1.3.C$_h$}$$

using the special structure. For if a special structure is advantageous for c, it ought to be so for any π. Then to combine a cut from (1.3.C)$_h$ with the remaining constraints $Dx \geq d$, one may simply take non-negative multiples, and in this way one would solve

$$\begin{aligned} &\max \quad \lambda d + \pi_0 \\ &\text{subject to} \quad \lambda D + \pi \leq c \\ &\qquad\qquad\quad \lambda \geq 0 \end{aligned} \tag{1.3.D}$$

(1.1.A) a disjunctive cut from (1.3.C)$_h$.

Indeed, $\lambda d + \pi_0$ is the right-hand-side of the combination of the two cuts, so one wants it to be as large as possible, since under the constraints of (1.3.D) the inequality $(\lambda D + \pi)x \geq \lambda d + \pi_0$ implies $cx \geq \lambda d + \pi_0$ (recall that $x \geq 0$).

It is true that (1.3.D) is easier than optimizing with the full disjunctive system (1.3.B)$_h$, but is it as good? Intuitively, there might be cases where it is not as good, because in (1.3.B)$_h$ one has the freedom of using a different multiplier $\lambda^h \geq 0$ on D in each (1.3.B)$_{h_i}$, while in (1.3.D) only the one multiplier $\lambda \geq 0$ is available.

This intuition turns out to be correct, and is one evident way of seeing why gaps can occur in Lagrangean duality. For it turns out that (1.3.D) is the Lagrangean dual problem, as we now show.

Assuming that either E is rational or that

$$T = \{x \mid Ex \geq e,\ x \geq 0 \text{ and integer}\}$$

is compact, clconv(T) will be a polyhedron, so that it has a definition by linear inequalities:

$$\text{clconv}(T) = \{x \geq 0 \mid Qx \geq q\}. \tag{1.3.E}$$

The disjunctive cuts (1.1.A) from the systems $(1.3.C)_t$ are precisely those valid for $\operatorname{clconv}(T)$ (see 1.1.2 above) hence those with $\pi = \theta Q$, $\pi_0 \leq \theta q$ for some $\theta \geq 0$, so that (1.3.D) is the linear program

$$\begin{aligned} \max \quad & \lambda d + \theta q, \\ \text{subject to} \quad & \lambda D + \theta Q \leq c, \qquad\qquad (1.3.F) \\ & \lambda \geq 0,\ \theta \geq 0, \end{aligned}$$

which is dual to the primal problem

$$\begin{aligned} \min \quad & cx \qquad\qquad (P^*) \\ \text{subject to} \quad & Dx \geq d \\ & x \in \operatorname{clconv}\{x \geq 0 \mid Ex \geq e,\ x \text{ integer}\} \end{aligned}$$

of Geoffrion [19] (recall (1.3.E) and the definition of T).

Next, since (P^*) is

$$\begin{aligned} \min \quad & cx \\ \text{subject to} \quad & Dx \geq d \qquad\qquad (1.3.G) \\ & Qx \geq q \\ & x \geq 0 \end{aligned}$$

by Lagrangean results for consistent linear programs, if (1.3.G) is bounded in value, it is equivalent to both

$$\max_{\lambda, \theta \geq 0} \min_{x \geq 0} \{cx + \lambda(d - Dx) + \theta(q - Qx)\}, \qquad (1.3.H)$$

and

$$\max_{\lambda \geq 0} \min \{cx + \lambda(d - Dx) \mid Qx \geq q,\ x \geq 0\}. \qquad (1.3.I)$$

Furthermore, in optima to (1.3.H) the optimal λ, θ (which exist) provide an optimal $\lambda = \bar{\lambda}$, $\pi = \bar{\theta} Q$ to (1.3.D), since $\bar{\lambda}, \bar{\theta}$ are optimal in the equivalent (1.3.F) to (1.3.D).

Finally, (1.3.I) is the Lagrangean dual, since

$$\begin{aligned} \min \quad & cx + \lambda(d - Dx) \\ \text{subject to} \quad & Ex \geq e \qquad\qquad (PR_\lambda) \\ & x \geq 0 \text{ and integer} \end{aligned}$$

is equivalent to

$$\begin{aligned} \min \quad & cx + \lambda(d - Dx) \qquad\qquad (1.3.J) \\ \text{subject to} \quad & x \in \{y \geq 0 \mid Qy \geq q\} \cap \operatorname{clconv}\{y \geq 0 \mid Ey \geq e,\ y \text{ integer}\} \end{aligned}$$

it one observes [19] that the infimum of a linear form on a set T is the infimum on $\operatorname{clconv}(T)$.

Fisher and Shapiro [16] have utilized the group problem [27] in (P), by in effect taking $Ex \geq e$ to be the convex span of the group points and taking $Dx \geq d$ to be the linear programming constraints; specifically, the group constraints are imposed in place of $Ex \geq e$. As the above analysis shows, they obtain the bounds from the polyhedron defined by intersecting the linear programming relaxation with the group polyhedron.

Here algebraic features, specifically the ease of enumerating irreducible elements of the group, allow one to bypass the explicit use of disjunctive constraints (1.5.C), in favor of a direct enumeration. The algorithm used in [16] is certainly not disjunctive in nature!

In this section, we saw that gaps occur in Lagrangian duality because the cuts of (1.3 B), from

$$(Dx \geq d \wedge Ex \geq e \wedge x = h) \vee (Dx \geq d \wedge Ex \geq e \wedge x = h') \vee \cdots$$

are generally more than those from the expression

$$(Dx \geq d) \wedge ((Ex \geq e \wedge x = h) \vee (Ex \geq e \wedge x = h') \vee \cdots),$$

where "$\wedge$" is "and" while "$\vee$" is "or". This means that the gaps occur because a distributive law of boolean logic fails in its cut formulation. In 2.1.2 we will give hypotheses that insure that distribution holds.

1.4. *Disjunctive cuts interpreted via subadditive functions*

Recall from [14] that a subset M of $\mathbf{R}^r$ is a monoid if $0 \in M$ and $v, w \in M$ implies $v + w \in M$ (i.e., M is an additive subgroup of $\mathbf{R}^r$). A function $f : M \to \mathbf{R} \cup \{+\infty\}$ defined on a monoid M is subadditive if

$$f(x + x') \leq f(x) + f(x') \quad \text{for } x, x' \in M. \tag{SUB}$$

The use of subadditive functions in cutting-plane theory originates in joint work of Gomory and Johnson and is continued and further developed in Johnson's researches; we discuss these contributions in [35, 36]. The subadditive functions used by Gomory and Johnson have the unit interval, modulo unity, as domain. They correspond to subadditive functions (SUB) with $M = \mathbf{R}^r$, which have unit periods in each coordinate direction (for details, see [34, Proposition 2.11]).

The basic principle of disjunctive cuts (1.1 above) can be cast in terms of subadditive functions. The first step is to determine the functions in terms of the space of the original variables x on which they are a certain subclass of the convex functions, then these are "transferred" to functions acting on the space of the columns of A in (IP), where they are subadditive, but generally not convex. Some of what follows is in [34].

A function f is called sublinear, if it is both subadditive and positively homogeneous:

$$
\begin{aligned}
&f(\lambda x) = \lambda f(x) \quad \text{for all } \lambda \geq 0 \\
&f(x + x') \leq f(x) + f(x')
\end{aligned} \tag{1.4.A}
$$

We called these functions conical in [34, 37] but it is best to use standard terminology when it exists. The sublinear functions are convex [45, 47], and include the gauge functions, for which one imposes $f(x) \geq 0$ in addition to (1.4.A).

For a subadditive function F, its directional derivative $\bar{F}(v)$ at zero in the direction v, is defined by

$$\bar{F}(v) = \limsup\{F(\delta v)/\delta \searrow 0+\}. \tag{DER}$$

In (DER), it is assumed that $\{\delta v \mid \delta \geq 0\}$ is in the domain of F. One easily proves that $\bar{F}$ is sublinear (see [37, 40]).

We recall from [34, 35, 36] that the set of all valid cuts for the general constraints

$$
\begin{aligned}
&Ax + By \in S \\
&x, y \geq 0 \quad (x = (x_1, \ldots, x_r), y = (y_1, \ldots, y_s)) \\
&x \text{ integer},
\end{aligned} \tag{GC}
$$

S a set, can be obtained via the general cut form

$$\sum_{j=1}^{r} F(a^{(j)})x_j + \sum_{k=1}^{s} \bar{F}(b^{(k)})y_k \geq \inf\{F(u) \mid u \in S\} \tag{CF}$$

where $a^{(j)}$ is the jth column of A and $b^{(k)}$ is the kth column of B, and F is a subadditive function with $F(0) = 0$.

To be precise, all cuts (CF) are valid for F subadditive with $F(0) = 0$; and if

$$\sum_{j=1}^{r} \pi_j x_j + \sum_{k=1}^{s} \sigma_k y_k \geq \pi_0 \tag{VC}$$

is valid, then there is a subadditive function F "behind" the cut (VC), in the sense that F satisfies:

(1) $\inf\{F(u) \mid u \in S\} \geq \pi_0$;
(2) $F(a^{(j)}) \leq \pi_j$, $j = 1, \ldots, r$;
(3) $\bar{F}(b^{(k)}) \leq \sigma_k$, $k = 1, \ldots, s$;
(4) $F(0) = 0$.

For full details, see [34, 35 or 36]. The pure-integer case of (CF), i.e., $s = 0$, is also given in [39].

Linear functions are of course sublinear, and one easily shows (e.g., [34, Prop. 2.1.]) that, if f_α is a class of sublinear functions indexed by a nonempty set I ($\alpha \in I$), and if $f(x) = \sup_\alpha f_\alpha(x)$ is everywhere finite on its domain, then f is sublinear. If $I \neq \emptyset$ is finite and all f_α are linear, we call f (homogeneous) polyhedral [45, 47].

The functions behind the disjunctive cuts (DC) are sublinear and if $H \neq \emptyset$ is finite, they are polyhedral. For put $f_h(x) = (\lambda^h A^h)x$, $f(x) = \sup_h f_h(x)$. $f(x)$ is sublinear, since it is the pointwise supremum of sublinear functions. Further, the jth intercept in (DC)$_j$ implies via (1.4.A) that $f(x)$ is finite on $x \geq 0$ (note

$$f(x) = f\Big(\sum_j x_j e_j\Big) \leq \sum_j x_j f(e_j) \quad \text{for all } x_j \geq 0),$$

hence sublinear. We rewrite the cut (DC) as

$$\sum_{j=1}^{r} f(e_j)x_j \geq \pi_0. \tag{1.4.B}$$

For any function f subadditive on the domain of all $x \geq 0$, x integer the following defines a function F on the set of all non-negative integer combinations of the columns of A in (IP):

$$F(v) = \inf\{f(x) \mid v = Ax,\ x \geq 0 \text{ and integer}\} \tag{1.4.C}$$

F is subadditive, for, if α resp. α' are strict upper bounds on $F(v)$ resp. $F(v')$ and x resp. x' are non-negative integer vectors with $v = Ax$ resp. $v' = Ax'$ and also $f(x) < \alpha$ resp. $f(x') < \alpha'$, then we have

$$\begin{aligned} F(v+v') &\leq f(x+x') \\ &\leq f(x) + f(x') \\ &< \alpha + \alpha' \end{aligned} \tag{1.4.D}$$

(1.4.D) follows since $v + v' = A(x + x')$ with $x + x' \geq 0$ integral, and since f is subadditive. By taking infima on the right in (1.4.D) on α and then α', we obtain $F(v + v') \leq F(v) + \alpha'$ and then the desired $F(v + v') \leq F(v) + F(v')$.

Now if f is the function of (1.4.B), with the disjunctive systems (S_h) derived from the constraints of (IP), whenever x is feasible in (IP) we have $f(x) \geq f_p(x) \geq \lambda^p b^p \geq \min_h \lambda^h b^h = \pi_0$, where $p \in H$ is any system (S_p) which holds for x. Therefore, $F(b) \geq \pi_0$ by the definition (1.4.C). It is also clear from (1.4.C) that $F(a^{(j)}) \leq f(e_j)$, $j = 1, \dots, r$, and hence (1.4.E) is obtained from the non-negativities $x \geq 0$ and

$$\sum_{j=1}^{r} F(a^{(j)})x_j \geq F(b). \tag{1.4.E}$$

This cut (1.4.E) is the pure integer case of (CF).

If some variables are continuous, i.e., in the general case (GC), the same analysis holds true. Since f is sublinear, for a continuous column $b^{(j)}$ and any $\delta > 0$ we have $F(\delta b^{(j)}) \leq f(\delta e_j) = \delta f(e_j)$, hence by the definition (DER), $\bar{F}(b^{(j)}) \leq f(e_j)$, and the form (CF) is obtained.

From (1.4.C) follows $F(Ax) \leq f(x)$. Functions G on the column space are transferred to the row-space of x by the formula

$$g(x) = G(Ax) \tag{1.4.F}$$

and therefore, if (1.4.C) is followed by (1.4.D), one ends back in the row space with $g(x) \leq f(x)$.

The reverse direction, that subadditive cuts (1.4.E) derive from disjunctive cuts (DC), is trivial: the subadditivity of F implies the validity of (1.4.E), hence there exist disjunctive multipliers $\lambda^h \geq 0$ which yield a cut (DC) on which (1.4.E) is a weakening.

The first interpretation of intersection/convexity cuts by subadditive functions is due to Burdet [10]. Specifically, with the hypotheses of the earlier principle given in the theorem of 1.2., plus the assumption that (IC) is valid for the group polyhedron, in either the pure-integer ($s = 0$) or pure-continuous ($r = 0$) cases, the cut (IC) is shown in [10] to be an instance of the general cut form (CF) with

$$F(u) = \inf\{\lambda > 0 \mid u/\lambda \in C\}. \tag{1.4.G}$$

Functions F of the form (1.4.G) are called gauge functions for the convex set C [45, 47], and they are gauge in the sense described above.

1.4.1. Abstract versus computational equivalence of cuts

Despite the theoretical interrelations above, the algebraic and the disjunctive methods are not equivalent, since there are distinctions which are not touched by these interrelations. For instance, to obtain a subadditive F from a sublinear f, the interrelation provides only F of (1.4.C), whose definition is in terms of a family of programs.

We ought to differentiate between: (1) Knowledge of properties of cuts and their interrelations; (2) Knowledge of which specific inequalities are valid cuts.

The interrelations given above are of type (1). Typically, a result contains knowledge of both types, as with the characterization of cuts by the constraints of (FDP) of [35, 36]. For type (1), we know that the valid cuts derive in a specific way from the constraints of (FDP). For type (2), we may devise methods for computing certain solutions to these constraints, and from these find certain specific cuts. If the constraint system is small, we may calculate all extreme solutions; otherwise, we may never actually know all the extreme solutions which, in theory, do exist.

The differentiation above, in terms of "knowledge," is inexact. To make it rigorous, one might differentiate classes of cuts according to the amount of computation they require. Cuts requiring extensive computation would not be valued, unless they are expected to be particularly effective. Similarly, an equivalence is primarily theoretical, if the reduction of one cut family to another requires a prohibitive computation.

1.4.2. A subadditive view of Lagrangean relaxations

The Lagrangean relaxations $(PR)_\lambda$ can be easily cast in terms of subadditive functions. Here fix $\lambda \geq 0$ and take $f(x) = (c - \lambda D)x$, and use (1.4.C) with $A = E$ to get the function on to column space.

Letting e_j be the jth column of E in $(\mathrm{PR})_j$, and $(\lambda D)_j$ the jth element of λD, the inequalities for the value function F of (1.4.C) become

$$F(e^j) + (\lambda D)_j \leq c_j, \quad j = 1, \ldots, r. \tag{1.4.2.A}$$

Again, we have a sum of valid cuts, one for clconv $\{x \geq 0 \mid Ex \geq e, x \text{ integer}\}$ (via F), one from $Dx \geq d$.

1.4.3. Improving disjunctive cuts by subadditive methods

Consider the constraints (GC) for a set $S \neq \emptyset$.

Suppose that a set M is known which is closed under addition and has $O \in M$ — i.e., a 'monoid' — with the property that

$$S + M \subseteq S. \tag{1.4.3.A}$$

(This is clearly equivalent to $S + M = S$, since $O \in M$.) In (1.4.3.A) we use the notation $S + M = \{s + m \mid s \in S, m \in M\}$.

Then for any $m^{(i)} \in M$, with $x, y \geq 0$ and x integer we see that (GC) implies

$$\sum_{i=1}^{r} (a^{(i)} - m^{(i)})x_i + \sum_{k=1}^{s} b^{(k)}y_k \in S + M \subseteq S \tag{1.4.3.B}$$

and therefore, from (CT),

$$\sum_{i=1}^{r} F(a^{(i)} - m^{(i)})x_i + \sum_{k=1}^{s} F(b^{(k)})y_k \geq \inf\{F(v) \mid v \in S\}. \tag{1.4.3.C}$$

Now in (1.4.3.C) the $m^{(i)}$ are arbitrary, so we conclude

$$\sum_{i=1}^{r} (\inf\{F(a^{(i)} - m) \mid m \in M\})x_i + \sum_{k=1}^{s} F(b^{(k)})y_k \geq \inf\{F(v) \mid v \in S\}. \tag{1.4.3.D}$$

This improves (CT).

The cut (1.4.3.D) appears in [7], where a version for bivalent variables x_i is also given. By setting $G(u) = \inf\{F(u - m) \mid m \in M\}$, one notes that (1.4.3.D) is implied by the general cut form for G. This provides an alternate proof of (1.4.3.D), but requires that one first prove the subadditivity of G; we leave this to the reader.

In typical applications, $Ax + By$ in (GC) is not the original constraints, but is the Simplex tableau, or part of it, perhaps filled out with unit rows:

$$z - b = Ax + By, \tag{1.4.3.E}$$

where z is certain of the variables, including some basic ones; x is the set of integer nonbasic variables; y is the continuous nonbasic variables; and b is the current solution of the linear programming relaxation. The variables z are picked, typically for the ability to exploit the constraints on them, say by disjunctive methods, for these constraints may be $z \in T$, with e.g.

$$T = \{z \mid z_i = 0 \text{ or } 1, i = 1, \ldots, p\}, \quad z = (z_1, \ldots, z_p).$$

Then S is picked to be a suitable relaxation of these constraints, with

$$S \supseteq \{v \mid v = z - b \text{ for some } z \in T\}. \tag{1.4.3.F}$$

The choosing of S is an art; S must be close enough to $\{v \mid v = z - b$ for some $z \in T\}$ so that fairly good disjunctive cuts are produced, but also (1.4.3.A) must hold for a monoid M which will allow the resulting strengthening (1.4.3.D) to be a good cut.

Note that, in the context discussed, with the disjunctive cut obtained from the condition $v \in S$, the resulting function of these cuts is in the variables v — "local variables," we say — and so the disjunctive function $f(v)$ is already on the column space of (DC). Hence we set $F = f$ in (1.4.3.D), and note that, since f is sublinear, $f^- = f^* = f$.

In typical situations also, S is determined by linear constraints, i.e.,

$$S = \{v \mid v = z - b, A'z \geq b', z \text{ integer}\} \tag{1.4.3.G}$$

with A', b' integral. Then the condition (1.4.3.B), with M a monoid, forces

$$M \subseteq \{z \mid A'z \geq 0, z \text{ integer}\} \tag{1.4.3.H}$$

so that, maximally, M contains the entire basis ((1.3.1.B) of [35, Part 1]) of $A'z \geq 0$ — in practice, one may utilize any of the solutions to $A'v \geq 0$, v integer, of which one is aware.

In all situations, it does not actually matter how one determines a lower bound π_0 with $\pi_0 \leq \inf\{F(v) \mid v \in S\}$; all one needs is to know that this holds, so any reasoning implied by $v \in S$ may be used to "design" F. The infima in (1.4.3.D) may or may not be easily calculated, but since all $\lambda_i \geq 0$, any quantity $F(a^{(i)} + m)$ for any $m \in M$ may be validly used as a cut coefficient.

The next section gives details on the use of (1.4.3.D).

1.4.3.1. *Non-redundancy of the strengthened cut.* This algebraic strengthening of the disjunctive constraints construction has a significant new property. Since (DC) is implied by at least one (S_h) holding, any branching scheme which imposes one of these systems (S_h) will make (DC) redundant. However, the strengthening (1.4.3.D) is generally not redundant after branching.

Because of this redundancy property of (DC), when used with an enumerative branching scheme one either chooses the systems (S_h) to be a different alternative than is branched on — in this manner obtaining some of the fathoming power of having used both types of branching — or else, when the branching alternative is (S_h), one uses (DC) only for penalty calculations. With the strengthening (1.4.3.D), these provisions can often be ignored.

1.5. *Some applications*

Here we provide a limited sampling of some of the situations to which the disjunctive methods have been applied. The papers [4, 6] demonstrate the versatility of these methods in several other contexts and supplement this section

Due to space limitations, we are forced to omit applications to problems with complementarity constraints (see e.g., [43]), separable programming [51], and the disjunctive facet problem and related topics (see e.g., [26]). See the earlier version of this paper for these other applications [35].

1.5.1. The fractional and mixed integer cuts of Gomory

From (1.4.3.F), if $\pi_0 \leq \inf\{F(v) \mid v \in S\}$ is implied simply by the condition that $v = z - b$ for z integer — i.e., if $A^1 = 0$ in (1.4.3.G) — then by (1.4.3.H) we may use $M = \{z \mid z \text{ integer}\}$ in the improved cut (1.4.3.D).

For instance, let the constraints

$$x_i = a_{i0} - \sum_{j \in J} a_{ij}(-t_j) \tag{TB}$$

express the basic variables x_i in terms of non-basic variables t_j of a Simplex Tableau whenever x_i is fractional in the current tableau. With local variable $v = x_i - a_{i0}$ we have the disjunctive constraints

$$v \leq -f_{i0} \quad \text{or} \quad v \geq 1 - f_{i0} \tag{1.5.1.A}$$

where f_{i0} is the fractional part of a_{i0}.

From (1.5.1.A) we obtain the function $f(v) = \max\{-\lambda_1 v, \lambda_2 v\}$ of one variable, as the function of disjunctive constraints ($\lambda_1, \lambda_2 \geq 0$). Putting $S = \{v \mid v = x_i - a_{i0}$ and x_i integer$\}$, we find

$$\begin{aligned} \inf\{f(v) \mid v \in S\} &= \min\{\lambda_1 f_{i0}, \lambda_2(1 - f_{i0})\} \\ &= 1, \end{aligned} \tag{1.5.1.B}$$

with the choice $\lambda_1 = 1/f_{i0}$, $\lambda_2 = 1/(1 - f_{i0})$.

Then (CF) is (recall $\bar{F} = f$)

$$\sum_{j \in J} \max\left\{\frac{a_{ij}}{f_{i0}}, \frac{-a_{ij}}{(1 - f_{i0})}\right\} t_j \geq 1. \tag{1.5.1.C}$$

Now letting J_1 resp. J_2 index the integral resp. the continuous non-basics, the strengthening (1.4.3.D) with $M = Z$ is

$$\sum_{j \in J_1} \min\left\{\frac{f_{ij}}{f_{i0}}, \frac{1 - f_{ij}}{1 - f_{i0}}\right\} t_j + \sum_{j \in J_2} \max\left\{\frac{a_{ij}}{f_{i0}}, \frac{-a_{ij}}{(1 - f_{i0})}\right\} t_j \geq 1. \tag{1.5.1.D}$$

This result (1.5.1.D) is an easy computation, which we omit. It is Gomory's mixed-integer cut [29]. If $J_2 = \emptyset$, (1.5.1.D) is a strengthening of Gomory's fractional cut [28]. For an alternate derivation, see [4].

For some other cuts for the mixed-integer group problem, with essentially the same derivation, take $p = 2$ rows of the tableau. In local variables $v_1 = x_1 - a_{10}$, $v_2 = x_2 - a_{20}$, the fact that $(x_1, x_2) \in Z^2$ implies that at least one of the disjunctive systems

$$v_1 \geq (1 - f_{10}), \quad v_2 \geq (1 - f_{20})$$

or

$$v_1 \geq (1 - f_{10}), \quad v_2 \leq -f_{20}$$

or

$$v_1 \leq -f_{10}, \quad v_2 \geq (1 - f_{20})$$

or

$$v_1 \leq -f_{10}, \quad v_2 \leq -f_{20}$$

holds. Here the disjunctive constraints function is

$$f(v_1, v_2) = \max\{\lambda_1 v_1 + \lambda_2 v_2,\ \delta_1 v_1 - \delta_2 v_2,\ -\tau_1 v_1 + \tau_2 v_2,\ -\theta_1 v_1 - \theta_2 v_2\} \tag{1.5.1.E}$$

with all eight parameters non-negative, and the constant term π_0 of the cut is

$$\begin{aligned} \min\{&\lambda_1(1 - f_{10}) + \lambda_2(1 - f_{20}),\ \delta_1(1 - f_{10}) + \delta_2 f_{20}, \\ &\tau_1 f_{10} + \tau_2(1 - f_{20}),\ \theta_1 f_{10} + \theta_2 f_{20}\} \end{aligned} \tag{1.5.1.F}$$

which is strictly positive unless both of some pair of parameters are set to zero.

For $j \in J$, clearly

$$\begin{aligned} &\inf_{m \in Z} f(a_{1j} + m_1, a_{2j} + m_2) \\ &= \max\left\{ f(w_1, w_2) \,\middle|\, \begin{array}{l} (w_1, w_2) = (f_{1j}, f_{2j}) \text{ or } (f_{1j} - 1, f_{2j}) \\ \text{ or } (f_{1j}, f_{2j} - 1) \text{ or } (f_{1j} - 1, f_{2j} - 1) \end{array} \right\} \end{aligned} \tag{1.5.1.G}$$

so the quantity on the right in (1.5.1.G) may be validly employed in place of the jth cut intercept (1.4.3.D).

1.5.2. *Set covering, set partitioning, and other logical constraints*

Suppose that a set-covering requirement

$$x_1 + \cdots + x_p \geq 1 \tag{1.5.2.A}$$

in bivalent variables x_j, $j = 1, \ldots, p$, either occurs among the constraints of (**MIP**), or is inferred from those constraints. Let the tableau rows for these x_j be given as in (TB) where unit rows have perhaps been adjoined.

Put $v = (v_1, \ldots, v_p)$, and in (1.4.3.G) put

$$S = \{v \mid v = x - b,\ x_1 + \cdots + x_p \geq 1,\ \text{all } x_j \text{ integer}\} \tag{1.5.2.B}$$

where $x = (x_1, \ldots, x_p)$, $b = (a_{10}, \ldots, a_{p0})$. Hence we find

$$M = \{x \mid x_1 + \cdots + x_p \geq 0,\ \text{all integer}\} \tag{1.5.2.C}$$

in (1.4.3.H).

One function which produces valid cuts for $v \in S$ is based on the disjunctive conditions for some $k = 1, \ldots, p$.

$$u_k \geq 1 - a_{k0} \tag{1.5.2.D}$$

and is

$$f(u) = \max\{\lambda_k u_k\} \quad (\lambda_k \geq 0). \tag{1.5.2.E}$$

Since (1.5.2.A) is insured by linear programming, only the integrality conditions on the x_k might be violated. If all $x_k < 1$, all $a_{k0} < 1$, and we have

$$\inf\{f(u) \mid u \in S\} = \min_k \lambda_k(1 - a_{k0}) > 0. \tag{1.5.2.F}$$

The strengthened cut (1.4.1.D) is

$$\begin{aligned} &\sum_{j \in J_1} \left(\inf_{m \in M} \max_k \{\lambda_k(-a_{kj} - m_k)\} \right) t_j \\ &\quad + \sum_{j \in J_2} \left(\max_k \{-\lambda_k a_{kj}\} \right) t_j \geq \min_k \lambda_k (1 - a_{k0}). \end{aligned} \tag{1.5.2.G}$$

This cut is one of those reported in [7], where an algorithm is stated for computing the coefficients of t_j, $j \in J_1$ in no more than $(p-1)$ elementary iterations. The process involves addition of the monoid basis elements for M.

Even when (1.5.2.A) is not a problem constraint, we may branch on it as a condition. On one branch one we impose the cut (1.5.2.G) implied by (1.5.2.A), and on the other we set $x_k = 0$ for all $k = 1, \ldots, p$. This partitioning is due to Balas [3].

Another application of (1.5.2.G) occurs in what we call "cross-branching" for bivalent variables, in which two fractional bivalent variables x_1, x_2 create a partition by the settings

$$x_1 = x_2 \quad \text{or} \quad x_1 = 1 - x_2 \tag{1.5.2.H}$$

The first condition $x_1 = x_2$ of (1.5.4.H) implies the two covering constraints

$$x_1 + x_2' \geq 1, \qquad x_1' + x_2 \geq 1 \tag{1.5.2.I}$$

where $x_1' = 1 - x_1$, $x_2' = 1 - x_2$.

We can obtain one cut (1.5.2.G) from each condition of (1.5.2.I), and similarly $x_1 = 1 - x_2$ implies two cuts.

For a set partitioning constraint

$$x_1 + \cdots + x_p = 1 \tag{1.5.2.J}$$

in bivalent variables, we have

$$M = \{x \mid x_1 + \cdots + x_p = 0\}$$

and using (1.5.2.J) and integrality of the x_j we can employ the disjunctive constraints in local variables:

for at least one pair (i, k).

$$v_i \geq 1 - a_{i0}, \qquad v_k \geq -a_{k0}. \tag{1.5.2.K}$$

These constraints provide the function

$$f(v) = \max\{\lambda_{ik}^1 v_i, \lambda_{ik}^2 v_k\} \qquad (\lambda_{ik}^1, \lambda_{ik}^2 \geq 0) \tag{1.5.2.L}$$

and the right-hand-side

$$\min_{i,k} \{\lambda_{ik}^1(1 - a_{i0}) - \lambda_{ik}^2 a_{k0}\} \tag{1.5.2.M}$$

in an improved cut (1.4.3.D).

If doing a project is represented in (MIP) by $w = 1$ for a bivalent variable w, then the fact that this project necessitates doing the projects represented by $x_1, \ldots, x_p$ is stated

$$w \leq x_k, \quad k = 1, \ldots, p, \tag{1.5.2.N}$$

which is put in the form (1.5.2.A) by using $w' = 1 - w$ and writing (1.5.2.N) as

$$1 \leq x_k + w', \quad k = 1, \ldots, p. \tag{1.5.2.O}$$

Similarly, conflicts between projects w and u become $w' + u' \geq 1$ with $u' = 1 - u$; the fact that w and u are alternatives becomes $w' + u' = 1$; the fact that w forces at least one of $x_1, \ldots, x_r$ to be done becomes $1 \leq w' + x_1 + \cdots + x_r$. These comments are simply by way of noting the importance of cuts derived from (1.5.2.A), to provide additional fathoming power both before and after branching is initiated.

1.5.3. "On-off switch" constraints

Geoffrion [19] points out the importance of constraints such as

$$\sum_{k=1}^{s} \beta_k y_k \leq \beta x \tag{1.5.3.A}$$

with x a zero-one variable and the y_k's continuous. Here all $\beta_k \geq 0$, $\beta > 0$, are integers.

These constraints arise in facility-location problems, or, more generally, where the bivalent variable x represents doing $(x = 1)$ or not doing $(x = 0)$ a project, and $y_1, \ldots, y_s$ are among the variables of a linear program which represent the activities of the project. The constraint (1.5.3.A) allows the doing of the project to "activate" all project variables, as well as serving as a means of expressing an economic restriction. These constraints supplement those (1.5.2.A) of the previous subsection, which can be used to represent the "pure logic" of the interrelations between projects.

Let x be currently fractional, represented as

$$x = a_0 + \sum_{j \in J} a_j(-t_j) \tag{1.5.3.B}$$

in the tableau, and let the y_k's be given by

$$y_k = b_{k0} + \sum_{j\in J} b_{kj}(-t_j) \tag{TB}$$

Setting $v = x - a_0$, $s_k = y_k - b_{k0}$ with f_0 the fractional part of a_0, we have the disjunctive constraints

$$\begin{aligned} &(v \geq 1 - f_0 \quad \text{and} \quad s_k \geq -b_{k0} \quad \text{for } k = 1, \dots, s) \\ &\quad \text{or } (v \leq -f_0 \quad \text{and} \quad s_k \geq -b_{k0} \quad \text{for } k = 1, \dots, s). \end{aligned} \tag{1.5.3.C}$$

From (1.5.3.C) we obtain the disjunctive constraints function

$$\begin{aligned} &f(v, s_1, \dots, s_s) = \\ &\quad \max\left\{\lambda^1 v - \sum_{k=1}^{s} \lambda_k^1 s_k, \ -\lambda^2 v - \sum_{k=1}^{s} \lambda_k^2 s_k\right\} (\lambda^1, \lambda^2, \lambda_k^1, \lambda_k^2 \geq 0) \end{aligned} \tag{1.5.3.D}$$

and the constant term

$$\min\left\{\lambda^1(1 - f_0) - \sum_{k=1}^{s} \lambda_k^1 b_{k0}, \ \lambda^2 f_0 - \sum_{k=1}^{s} \lambda_k^2 b_{k0}\right\}. \tag{1.5.3.E}$$

Since the non-negativity of the tableau rows for the y_k was used in (1.5.3.D), (1.5.3.E), the cut is based upon the set

$$S = \left\{(x, y_1, \dots, y_s) \,\middle|\, \begin{array}{c} -\beta_0 x + \sum_{k=1}^{s} \beta_k y_k \geq 0 \\ \text{all } y_k \geq 0 \end{array}\right\} \tag{1.5.3.F}$$

Its elements include

$$(\beta_1, \beta_0, 0, \dots, 0), (\beta_2, 0, \beta_0, 0, \dots, 0), \dots, (\beta_s, 0, \dots, \beta_0),$$

any of which, in non-negative integer combinations, can be used to strengthen the basic disjunctive cut.

2. Some theoretical aspects of the disjunctive approach

The theory associated with the disjunctive methods is more recent, and consequently less extensive, than that associated with the algebraic methods. We discuss some topics which have been treated at this writing.

We shall summarize several results obtained in [34, 37] and give some new ones. Our treatment is in terms of "en-propositions." We use the en-propositions to deal with logical conditions stated in arbitrary form, and as a setting in which to discuss theoretical issues, such as exactness and distributivity, whose importance in the context of the disjunctive normal form (S.) has already been established.

Balas' study in [5], in terms of the disjunctive normal form (S_∞), contains several important results for which we have not found any essentially simpler proofs. In particular, [5] contains necessary and sufficient conditions for an inequality to be a facet of the convex hull of feasible points, based on polars and reverse polars of arbitrary sets, and [5] discusses ways of computing facets by linear programming. Another important result of [5], regarding a distributivity relation, is cited in Section 2.1.2 below and generalized there.

2.1. *Construction of co-propositions*

To develop the systematic application [35, 37] of the ideas implicit in the principle of 1.1 above, we consider a propositional logic [18, 44] in atomic letters $P, Q, R, \ldots$ with propositions denoted $A, B, C, \ldots$. The atomic propositions will always stand for a linear inequality assertion

$$a_1x_1 + \cdots + a_nx_n \geq a_0 \tag{2.1.A}$$

and more complex propositions are constructed by putting '$\vee$' (for: "or") or '$\wedge$' (for: "and") between two given propositions, where $B \vee D$ allows for the possibility that both B and D are true.

To every proposition A, we inductively assign a co-proposition $\mathrm{CT}(A)$, which is a polyhedral cone of cuts (1.1.A) that are valid if A is true (we change terminology from [37]).

Ignore, for the moment, the ground step in the inductive assignment of the co-proposition $\mathrm{CT}(P)$ to the proposition P. (The ground step changes, depending on whether or not $x \geq 0$.) Two inductive rules clearly are suggested by the concept of a co-proposition, as vaguely as it has been described above.

The first rule is

$$\mathrm{CT}(B \wedge D) = \mathrm{CT}(B) + \mathrm{CT}(D). \tag{2.1.B}$$

Indeed if all cuts of $\mathrm{CT}(B)$ are valid when B holds, and all cuts of $\mathrm{CT}(D)$ are valid when D holds, then when $B \wedge D$ holds all cuts of $\mathrm{CT}(B) \cup \mathrm{CT}(D)$ are valid, and valid cuts are closed under addition.

The second rule is

$$\mathrm{CT}(B \vee D) = \mathrm{CT}(B) \cap \mathrm{CT}(D). \tag{2.1.C}$$

Indeed if $B \vee D$ is true, but we do not know which, all we are certain of is that those cuts which are valid on account of either are true; and (2.1.C) expressses this fact.

Clearly, $\mathrm{CT}(A)$ will depend on the syntactic form of A, as well as the truth set of A, because in general

$$\begin{aligned}\mathrm{CT}(B \vee (D_1 \wedge D_2)) &= \mathrm{CT}(B) \cap (\mathrm{CT}(D_1) + \mathrm{CT}(D_2)) \\ &\neq (\mathrm{CT}(B) \cap \mathrm{CT}(D_1)) + (\mathrm{CT}(B) \cap \mathrm{CT}(D_2)) \\ &= \mathrm{CT}((B \vee D_1) \wedge (B \vee D_2)).\end{aligned}$$

The ground step of the induction is also easy. If P is atomic and asserts (2.1.A), then (1.1.A) is implied by (2.1.A) and $x \geq 0$ precisely if there is a scalar $\lambda \geq 0$ with

$$
\begin{aligned}
&\lambda a_j \leq \pi_j, \quad j = 1, \ldots, r \\
&\lambda a_0 \geq \pi_0.
\end{aligned}
\tag{2.1.D}
$$

Now we will always take a valid cut (1.1.A) in the homogeneous form

$$\pi_0 x_0 + \sum_{j=1}^{r} (-\pi_j) x_j \leq 0 \tag{2.1.E}$$

and therefore a possible assignment of a cone $CT(P)$ of valid cuts to P is:

$$
\begin{aligned}
CT(P) = \operatorname{cone}\{&(a_0, -a_1, \ldots, -a_r), (-1, 0, \ldots, 0), \\
&(0, -1, 0, \ldots, 0), \ldots, (0, \ldots, 0, -1)\}
\end{aligned}
\tag{2.1.F}
$$

since this cone includes all $(\pi_0, -\pi_1, \ldots, -\pi_r)$ for which some $\lambda \geq 0$ exists with (2.1.D). Here, cone (S) is the smallest closed convex cone containing S. Since $CT(P)$ has a finite basis, it is polyhedral [45, 47, 49].

The rules (2.1.B) and (2.1.C) do not depend on the ground step (2.1.F). Indeed, any cone of cuts valid for the inequality (2.1.A) with all x_j integer could have been used: (2.1.F) is obtained without invoking the integrality of x. More generally, any valid cone of cuts $CT(B)$ can be used in these inductive assignments with B occurring as a well-formed proposition that is part of the proposition A for which a $CT(A)$ is desired.

For any proposition B, all of whose linear inequalities are rational, the set of all of the valid implied cuts (1.1.A) for $x \geq 0$ with x integer, is a polyhedral cone. Indeed, B can be expressed as a disjunctive system (S_h) with H finite: one considers all the possible combinations of "true" or "false" for the atomic letters (2.1.A) occurring in B, and by listing all the combinations which make B true, and placing "or" between them, the systems (S_h) are obtained. For each system (S_h), since A^h and b^h are assumed rational, the set of all integral solutions is either empty or a set with convex span a polyhedron, so the set of all implied inequalities is a polyhedron Γ_h. Therefore the set of all inequalities validly implied by B itself is the polyhedron $\Gamma = \bigcap_{h \in H} \Gamma_h$.

In practice, for $CT(P)$ one takes any polyhedral cone of valid cuts (1.1.A), usually a cone lying between that of (2.1.F) and Γ of the last paragraph. Then inductively by (2.1.B), $CT(B \wedge D)$ is a polyhedral cone: given finite bases for each of $CT(B)$ and $CT(D)$, their union is a finite basis for $CT(B \wedge D)$. Inductively by (2.1.C), $CT(B \vee D)$ is a polyhedral cone: it is defined by imposing all the defining inequalities for both $CT(B)$ and $CT(D)$. Hence $CT(A)$, for any proposition A, will be a polyhedral cone.

The reader desiring a discussion of polyhedra, bases for polyhedra, and polarity for polyhedra, may wish to consult [45, 47, Chapter 2] and the original paper [49].

Theorem. *Suppose that the proposition A states that at least one of the systems (S_h), $h \in H$ (H finite) hold in $\mathbb{R}^n$. That is, A states*

$$(A^1x \geq b^1) \vee \cdots \vee (A^t x \geq b^t) \tag{2.1.G}$$

where $H = \{1, \dots, t\}$ and a matrix inequality $Cx \geq f$ abbreviates the conjunction (i.e., repeated use of '$\wedge$') of the individual inequalities. Then $CT(A)$ consists precisely of all cuts (DC) of Section 1.1, when (2.1.F) is used for atomic letters.

Proof. In $CT(A^h x \geq b^h)$ are all cuts (1.1.A) with $\pi \geq \lambda^h A^h$ and $\pi_0 \leq \lambda^h b^h$ for any $\lambda^h \geq 0$, as one sees by repeated application of (2.1.B). Then deriving $CT(A)$ by repeated intersection as in (2.1.C) amounts precisely to taking the maxima indicated in (DC). **Q.E.D.**

Incidentally, the inductive clauses (2.1.B) and (2.1.C) also yield Balas' disjunctive constraint cut [5] for $x \geq 0$ deleted in (S_h), when the ground step of the induction is changed to

$$CT(P) = \text{cone}\{(a_0, -a_1, \dots, -a_n), (-1, 0, \dots, 0)\}. \tag{2.1.F$'$}$$

Since $CT(A)$ represents valid cuts deduced from the fact that A is true, there is a natural problem relaxation $cp(A)$ associated with A, which consists of all $x \in \mathbb{R}^n$ satisfying all cuts (1.1.A) of $CT(A)$. That is,

$$cp(A) = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{c} \sum \pi_j x_j \geq \pi_0 \text{ whenever} \\ (\pi_0, -\pi_1, \dots, -\pi_n) \in CT(A) \end{array} \right\}. \tag{2.1.H}$$

The condition $x \geq 0$ can be appended in $cp(A)$ when (2.1.F) is used, or any cone of cuts which includes, along with a given cut, all the weakenings of that cut, as obtained by use of the unit vectors $(-1, 0, \dots, 0)$, $(0, -1, 0, \dots, 0), \dots, (0, 0, \dots, -1)$ of (2.1.F). The unit vector $(-1, 0, \dots, 0)$ of (2.1.F) is always validly included even when the variables x are not non-negative.

The following result is easily proven from the standard facts concerning polarity of polyhedra and we omit the proof (for a proof, see [35]).

Lemma. *If P is the atomic sentence (2.1.A) and the ground step is (2.1.F), then*

$$cp(P) = \left\{ x \geq 0 \;\middle|\; \sum_{j=1}^{n} a_j x_j \geq a_0 \right\}. \tag{2.1.I}$$

With ground step (2.1.F)$'$,

$$cp(P) = \left\{ x \;\middle|\; \sum_{j=1}^{n} a_j x_j \geq a_0 \right\}. \tag{2.1.J}$$

$cp(A)$ does provide a problem relaxation, in the very definite sense of the next result. We use the notation $A(x)$ to emphasize the dependence of A on x.

Theorem. *If (2.1.J') is the ground step,*

$$\mathrm{cp}(A) \supseteq \mathrm{clconv}\{x \geq 0 \mid A(x) \text{ is true}\}. \tag{2.1.K}$$

For the ground step (2.1.F)',

$$\mathrm{cp}(A) \supseteq \mathrm{clconv}\{x \mid A(x) \text{ is true}\}. \tag{2.1.L}$$

Proof. Use (2.1.I) resp. (2.1.J) for the ground steps of an induction, and the inductive step is possible by (2.1.B), (2.1.C), and (2.1.H). Q.E.D.

We remark that equality rarely holds in (2.1.K) or (2.1.L), with the possible exception that A is in disjunctive normal form or that A has some other special property (see, e.g., Section 2.1.2 below).

Theorem. *If* $\mathrm{cp}(A)$ *is fully-dimensional, then the faces of* $\mathrm{cp}(A)$ *are precisely the extreme rays of* $\mathrm{CT}(A)$ *except possibly for a ray* $(-1, 0, \ldots, 0)$.

Proof. Omitted, since it easily follows from a knowledge of polarity for polyhedra (or see [35] for a proof). Q.E.D.

Whenever $\mathrm{clconv}\{x \mid A(x) \text{ is true}\}$ is fully-dimensional, as occurs in a fully-dimensional integer program, (2.1.K) shows that $\mathrm{cp}(A)$ is fully dimensional.

2.1.1. Exactness for co-propositions

From the nature of the reasoning behind (2.1.B), one expects

$$\mathrm{cp}(B \wedge D) = \mathrm{cp}(B) \cap \mathrm{cp}(D). \tag{2.1.1.A}$$

(2.1.1.A) is in fact true and easy to prove (see, e.g., [35] or [37, p. 88]).

From the same intuitions, one expects also $\mathrm{cp}(B \vee D) = \mathrm{clconv}(\mathrm{cp}(B) \cup \mathrm{cp}(D))$, a condition we call the exactness of $B \vee D$. Exactness may fail, basically for the same reason that consistency is needed in one of the converses of the disjunctive constraints principle of 1.1: an example of its failure is in [37]. But exactness does hold under so many broadly defined circumstances, that it rarely fails in connection with applications to (IP).

To explore the issue of exactness, assume a general situation in which non-negativities are not necessarily tacitly added to all atomic inequalities (2.1.A) — i.e., assume a situation like (2.1.F)', as opposed to (2.1.F). Then let a polyhedral definition

$$\mathrm{CT}(B_s)^\circ = \{x \mid Q^s x - q^s x_0 \geq 0,\ x_0 \geq 0\}$$

be given for $\mathrm{CT}(B_s)^\circ$, where S° denotes the polar set to $S \subseteq \mathbf{R}^{n+1}$, and where $x_0 \geq 0$ can always be appended due to the ability to indefinitely decrease π_0 in any valid cut (1.1.A). One easily shows that $\mathrm{cp}(B_s) = \{x \mid Q^s x \geq q^s\}$ (see e.g., [35]).

Note that the definition (2.1.C) for $\mathrm{CT}(B_1 \vee \cdots \vee B_t)$, modified in the obvious manner for $t \geq 3$, amounts to the following when (2.1.1')' is used. We have, by Farkas' Lemma and standard properties of polarity,

$$
\begin{aligned}
\mathrm{CT}(B_h) &= \mathrm{CT}(B_h)^P \\
&= \{x \mid Q^h x - q^h x_0 \geq 0,\ x_0 \geq 0\}^P \\
&= \left\{(\pi_0, -\pi) \,\middle|\, \begin{array}{c}\text{for some vector } \lambda^h \geq 0 \text{ and scalar } \sigma_h \geq 0, \\ (\pi, -\pi_0) = \lambda^h[Q^h, -q^h] + \sigma_h[0, 1]\end{array}\right\}, \\
&= \left\{(\pi_0, -\pi) \,\middle|\, \begin{array}{c}\pi = \lambda^h Q^h \text{ and } \pi_0 \leq \lambda^h q^h \\ \text{for some vector } \lambda^h \geq 0\end{array}\right\}.
\end{aligned}
$$

Hence one has (1.1.A) as a cut in $\mathrm{CT}(B_1 \vee \cdots \vee B_t)$ precisely if there is a vector $\lambda^h \geq 0$ with $\pi = \lambda^h Q^h$ and $\pi_0 \leq \lambda^h q^h$ for $h = 1, \ldots, t$.

Reviewing our reasoning of 1.1.1 above, regarding the general hypothesis (1.1.1.B) for the converse to the disjunctive cut principle, we see that it applies here as well.

Theorem. *If some* $\mathrm{cp}(B_h) \neq \emptyset$ *and if, for every* h *with* $Q^h x \geq q^h$ *inconsistent, we have*

$$Q^h x \geq 0 \implies x = \sum \{x^{(p)} \mid Q^p x \geq q^p \text{ consistent}\} \tag{2.1.1.B}$$

for certain $x^{(p)}$ *with* $Q^p x^{(p)} \geq 0$, *then* $\mathrm{CT}(B_1 \vee \cdots \vee B_t)$ *includes all valid cuts for* $\mathrm{clconv}\,(\bigcup_{h=1}^{t} \{x \mid Q^h x \geq q^h\})$. *Also*

$$\mathrm{clconv}\left(\bigcup_{h=1}^{t} \{x \mid Q^h x \geq q^h\}\right) = \mathrm{cp}(B_1 \vee \cdots \vee B_t)$$

and $\mathrm{CT}(B_1 \vee \cdots \vee B_t)$ *is exact.*

Proof. Omitted; for more details see [35]. Q.E.D.

A second, narrower, hypothesis insuring exactness is that all $\mathrm{cp}(B^h) = \emptyset$, i.e., all the systems $Q^h x \geq q^h$ are inconsistent. We omit the proof (see [37] for a proof).

For two particular applications of the above theorem, we have the following result.

Proposition. *Exactness holds if either:*

(1) *All* $Q = Q^h$, *independent of* $h = 1, \ldots, t$ *(see [37]);*

(2) $\mathrm{cp}(B_1 \vee \cdots \vee B_t)$ *is bounded.*

Proof. For (2), note that the general relation [36]

$$\mathrm{cp}(B_1 \vee \cdots \vee B_t) \supseteq \mathrm{clconv}\,(\mathrm{cp}(B_1) \cup \cdots \cup \mathrm{cp}(B_t)) \tag{2.1.1.C}$$

always holds. Clearly, (2.1.1.C) handles the case $\mathrm{cp}(B_1 \vee \cdots \vee B_t) = \emptyset$. For $\mathrm{cp}(B_1 \vee \cdots \vee B_t) \neq \emptyset$, at least one $\mathrm{cp}(B_k) \neq \emptyset$, i.e., $Q^k x \geq q^k$ is consistent. To obtain (2.1.1.B) it suffices, therefore, to show that

If $Q^i x \geq q^i$ is inconsistent, then $Q^i x \geq 0$ implies $x = 0$. (2.1.1.D)

Toward (2.1.1.D), let $x^0 \neq 0$ be given with $Q^i x^0 \geq 0$, and let x^* be such that $Q^k x^* \geq q^k$. Now if $(\pi_0, \pi_1, \ldots, \pi_r) \in \mathrm{CT}(B_i)$, we have $\pi x^0 \geq 0$ with $\pi = (\pi_1, \ldots, \pi_r)$:

$$\pi x^0 = \lambda^i Q^i x^0 \text{ (since } \pi = \lambda^i Q^i\text{)}$$

$$\geq 0 \text{ (since } Q^i x^0 \geq 0,\ \lambda^i \geq 0\text{)}.$$

Therefore $\pi x^0 \geq 0$ if

$$(\pi_0, \pi) \in \mathrm{CT}(B_1 \vee \cdots \vee B_t) = \mathrm{CT}(B_1) \cap \cdots \cap \mathrm{CT}(B_t) \subseteq \mathrm{CT}(B_i).$$

Also, if $(\pi_0, \pi) \in \mathrm{CT}(B_1 \vee \cdots \vee B_t) \subseteq \mathrm{CT}(B_k)$, then $\pi x^* \geq \pi_0$. Therefore, for any $\lambda \geq 0$, $\pi(x^* + \lambda x^0) \geq \pi_0$, showing that $\mathrm{cp}(B_1 \vee \cdots \vee B_t)$ is unbounded. This contradiction gives (2.1.1.D), and the proof of (2) is complete. Q.E.D.

This whole analysis can be repeated with the ground step (2.1.F) and with the same results obtained: one simply appends $x \geq 0$ to the inequalities $Q^i x - q^i x_0 \geq 0$, $x_0 \geq 0$.

2.1.2. *Distributivity for cp-propositions*

The polyhedral sets of $\mathbf{R}^r$, while they do have the lattice structure of a greatest lower bound for two sets (take intersection) and a least upper bound for two sets (take the closed, convex span of their union), do not form a distributive lattice: the distributive law

$$F_1 \cap \mathrm{clconv}(F_2 \cup F_3) = \mathrm{clconv}((F_1 \cap F_2) \cup (F_1 \cap F_3))$$

often fails, as we see in $r = 1$ taking $F_1 = \{1\}$, $F_2 = \{0\}$, $F_3 = \{2\}$. However, the truth value of $A \wedge (B \vee D)$ is that of $(A \wedge B) \vee (A \wedge D)$, i.e., the $\wedge$, $\vee$ — subpart of propositional logic is a distributive lattice. This asymmetry in the two lattices causes the mapping $A \to \mathrm{cp}(A)$ to depend on the syntactic form of A as much as the truth set of A.

"Half" of the distributive laws do hold in the relaxations $\mathrm{cp}(A)$.

$$\mathrm{cp}(B \wedge (D_1 \vee \cdots \vee D_t)) \supseteq \mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t)) \qquad (2.1.2.A)$$

$$\mathrm{cp}(B \vee (D_1 \wedge \cdots \wedge D_t)) \subseteq \mathrm{cp}((B \vee D_1) \wedge \cdots \wedge (B \vee D_t)). \qquad (2.1.2.B)$$

These laws, and several others, are established in [37].

From the inductive definitions (2.1.B), (2.1.C) and the ground clause (2.1.F) which introduces a parameter $\lambda \geq 0$ (a different parameter for each occurrence of P), more parameters are required for $\mathrm{CT}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t))$ than for

$\mathrm{CT}(B \wedge (D_1 \vee \cdots \vee D_t))$, showing that the former may include better cuts (see (2.1.2.A)), but these generally require more computation than those of the latter. Indeed, there are several interrelations between $\mathrm{cp}(A)$ and the number of parameters needed for $\mathrm{CT}(A)$, e.g.,

$$\mathrm{cp}((Ax \geq b) \wedge (x_1 = 0 \vee x_1 = 1) \wedge \cdots \wedge (x_n = 0 \vee x_n = 1))$$

is usually much larger than $\mathrm{cp}(\vee_{h \in H}(S_h)')$ with $(S_h)'$ from 1.1. However, (2.1.2.B) shows that, in some cases, more parameters can be worse.

In [5] Balas found an hypothesis on polyhedra, in order for the distributive law

$$F \cap \mathrm{clconv}\left(\bigcup_{h=1}^{t} F_h\right) = \mathrm{clconv}\left(\bigcup_{h=1}^{t} (F \cap F_h)\right) \tag{2.1.2.C}$$

to be valid. The hypothesis is that $F = \{x \mid ax = b\}$ where $ax \geq b$ is a face (possibly empty) of the bounded set $\mathrm{clconv}(\bigcup_{h=1}^{t} F_h)$.

From (2.1.2.C), the set of all valid cutting-planes for the left-hand-side of (2.1.2.C) is the set of valid cutting-planes for the right-hand-side. However, without further analysis, the co-propositions corresponding to the left-hand-side and the right-hand-side of (2.1.2.C) need not be equal, since the co-propositions are only some of the valid inequalities for a given set, and which ones they are depend on how the set is described.

Nevertheless, a co-propositional form of Balas' result is valid, and we give it next.

Theorem. *If* $\mathrm{cp}(B) \cap \mathrm{cp}(D_1 \vee \cdots \vee D_t)$ *is a face of* $\mathrm{cp}(D_1 \vee \cdots \vee D_t)$, *and* $\mathrm{cp}(D_1 \vee \cdots \vee D_t)$ *is bounded, then*

$$\begin{aligned} \mathrm{cp}(B \wedge (D_1 \vee \cdots \vee D_t)) &= \mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t)) \\ &= \mathrm{clconv}\left(\bigcup_{h=1}^{t} (\mathrm{cp}(B) \cap \mathrm{cp}(D_h))\right). \end{aligned} \tag{2.1.2.D}$$

Proof. Part of (2.1.2.D) is easy, since the boundedness of

$$\mathrm{cp}(D_1 \vee \cdots \vee D_t) \supseteq \mathrm{cp}(B \wedge (D_1 \vee \cdots \vee D_t)) \supseteq \mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t))$$

shows the boundedness, hence the exactness, of $\mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t))$ from the proposition in 2.1.1 above; this is one equation of (2.1.2.D).

For the remaining equation, note that any face of the bounded set

$$\mathrm{cp}(D_1 \vee \cdots \vee D_t) = \mathrm{clconv}\left(\bigcup_{h=1}^{t} \mathrm{cp}(D_h)\right) = \mathrm{conv}\left(\bigcup_{h=1}^{t} \mathrm{cp}(D_h)\right)$$

is the convex span of those points of the generating set $\bigcup_{h=1}^{t} \mathrm{cp}(D_h)$ which lie in it — again, exactness here is implied by boundedness.

Suppose that $x \in \mathrm{cp}(B \wedge (D_1 \vee \cdots \vee D_t)) = \mathrm{cp}(B) \cap \mathrm{cp}(D_1 \vee \cdots \vee D_t)$. By the last paragraph, since $\mathrm{cp}(B) \cap \mathrm{cp}(D_1 \vee \cdots \vee D_t)$ is a face of $\mathrm{cp}(D_1 \vee \cdots \vee D_t)$, there is a representation

$$x = \sum_{k=1}^{t} \lambda_k x^{(k)}, \qquad \sum_{k=1}^{t} \lambda_k = 1, \tag{2.1.2.F}$$

$\lambda_k \geq 0$ for $k \in H$; $\lambda_k = 0$ if $\mathrm{cp}(D_k) \cap \mathrm{cp}(B) = \emptyset$; in which $x^{(k)}$ is in $\mathrm{cp}(D_k)$ and in the face $\mathrm{cp}(B) \cap \mathrm{cp}(D_1 \vee \cdots \vee D_t)$. Hence each $x^{(k)} \in \mathrm{cp}(B) \cap \mathrm{cp}(D_k)$, yielding by (2.1.2.E) that

$$x \in \mathrm{clconv}\left(\bigcup_{k=1}^{t} (\mathrm{cp}(B) \cap \mathrm{cp}(D_k))\right) = \mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t)).$$

This shows that

$$\mathrm{cp}(B \wedge (D_1 \vee \cdots \vee D_t)) \subseteq \mathrm{cp}((B \wedge D_1) \vee \cdots \vee (B \wedge D_t)).$$

and (2.1.2.A) supplies the reverse inclusion. Thus the remaining equality of (2.1.2.D) is proven. Q.E.D.

2.1.3. *Linear programs equivalent to disjunctive systems*

For the disjunctive systems (S_H) with $|H|$ finite, the disjunctive inequalities $\pi x \geq \pi_0$ arise from the projection of the polyhedron

$$\begin{aligned} \lambda^h A^h &\geq \pi, \\ \lambda^h b^h &\geq \pi_0, \quad (\text{all } h \in H), \\ \lambda^h &\geq 0, \end{aligned} \tag{2.1.3.A}$$

upon the $(r+1)$ co-ordinates of (π, π_0). This is simply the principle in 1.1 of disjunctive cuts, and in 2.1 above we saw that this projection gives $\mathrm{Cl}((A^1x \geq b^1) \vee \cdots \vee (A^t x \geq b^t))$, $H = \{1, \ldots, t\}$, under (2.1.1). Assume this computation is exact.

Therefore, from 2.1, if the disjunctive systems (S_H) describe a fully-dimensional body, the facets of $\mathrm{clconv}\{x \mid \text{for some } h \in H,\ A^h x \geq b^h \text{ and } x \geq 0\}$ arise as certain of the projections upon the co-ordinates (π, π_0), of the extreme rays of the polyhedral cone of all solutions $(\{\lambda^h \mid h \in H\}, \pi, \pi_0)$ to (2.1.3.A) [5].

If one wishes to

$$\begin{aligned} &\text{minimize } cx \\ &\text{subject to } A^h x \geq b^h,\ x \geq 0 \quad \text{for some } h \in H, \end{aligned} \tag{2.1.3.B}$$

then one approach is to find the best disjunctive cut with $\pi = c$. This gives

$$\begin{aligned} &\max\ \pi_0 \\ &\text{subject to } \lambda^h A^h \leq c \\ &\qquad \lambda^h b^h \geq \pi_0 \quad (\text{all } h \in H) \\ &\qquad \lambda^h \geq 0. \end{aligned} \tag{2.1.3.C}$$

The ordinary linear programming dual to (2.1.3.C) is

$$\begin{aligned}
&\min \sum_{h=1}^{t} cx^h \\
\text{subject to}\quad & A^h x^h - b^h x_h \geq 0, \quad h \in H \\
& \sum_{h=1}^{t} x_h = 1 \\
& x^h, x_h \geq 0, \quad h \in H,
\end{aligned} \tag{2.1.3.D}$$

where, for each $h \in H$, we have introduced an n-vector x^h and a scalar x_h. (2.1.3.D) provides a linear programming equivalent to the purely logical program (2.1.3.B), under the same circumstances that the disjunctive cuts provide all valid cuts for (2.1.3.B), e.g., when all the systems (S_h), $h \in H$, are consistent (see [5]), or the associated co-proposition is exact.

One can develop a linear programming formulation for every proposition A in $\forall$, $\wedge$, such that the linear program describes optimization over $\mathrm{cp}(A)$ [37]. When specialized to (2.1.3.B), this program is (2.1.3.D).

Clearly, one way to compute the optimum $\bar{z}$ of (2.1.3.D) is to separately find the values $\bar{z}_h = \min\{cx \mid A^h x \geq b^h, x \geq 0\}$ and put $\bar{z} = \max \bar{z}_h$. This corresponds to putting $x_k = 1$ in (2.1.3.D) for one index k with $\bar{z} = \bar{z}_k$, and x^k equal to the optimum solution yielding $\bar{z}_k$. This method corresponds to the obvious branch-and-bound procedure for solving (2.1.3.B). In this way also, using systems like $(S_h)'$ for (S_h), one recovers ordinary branch-and-bound as one specific way of implementing (2.1.3.D).

For a more detailed discussion of generalized branch-and-bound schemes and their relations to disjunctive cuts that have the redundancy property cited in 1.4.3.3 above, see [37, Section 5].

2.1.4. *For future research*

By leaving open the exact nature of $CT(B)$ for propositions B "not further analysed," so long as (2.1.B), (2.1.C) are used inductively to determine other co-propositions, we are of course allowing for an improvement by algebraic means, in the broad sense that "algebraic" is used in [35, 36].

For if $x \in \mathrm{clconv}\{y \mid B(y)$ is true$\}$ implies $Qx \geq q$ with Q, q rational, then any slice term (1.3.2.M) of [35, Part 1] gives rise to the polyhedral cone of all (π, π_0), $\pi = (\pi_1, \ldots, \pi_n)$, with

$$\begin{aligned}
&\pi p^{(i)} \geq \pi_0, \quad i = 1, \ldots, a, \\
&\pi x^{(j)} \geq 0, \quad j = 1, \ldots, b.
\end{aligned} \tag{2.1.4.A}$$

As before, the practical use of (2.1.4.A) depends on designing $Qx \geq q$ to allow efficient descriptions (2.1.4.A).

The current understanding of how to properly devise "efficient relaxations" $Qx \geq q$ of logical conditions H, is poor. On the one hand, we have the example of group relaxations, as for instance the group of [27]: here the irreducible group elements allow efficient enumeration of $e^{(i)}$ for cutting-plane purposes (in smaller groups), and the inequalities $\pi x^{(i)} \geq 0$ become simple non-negativities. On the other hand, we have the principle of 2.4.3 above, in which the directions of infinity $x^{(i)}$ give the monoid M that allows cut-strengthening.

Clearly, we need more instances of "efficient relaxations" to understand the phenomenon better. Interestingly enough, those relaxations which have given us some very good cuts are unbounded, even though virtually all practical integer programs are bounded. As regards the set of $x^{(i)}$ of (2.1.4.A), one expects the presence of certain automorphisms of this set to yield an "efficient relaxation", but the sense of this certainly needs clarification.

2.2. Finitely-convergent disjunctive cutting-plane algorithms

We apply the results of 2.1.1 to obtain finiteness proofs for a class of cutting-plane algorithms for problems involving both linear and logical constraints; these problems include bounded (IP).

In principle, the use of systems $(S_n)^J$ solves (IP) by cutting-planes in one application; however, the computation of the cutting plane may be the work of a partial enumeration to solve (IP). The individual cutting-planes added at each iteration must be much simpler than those from $(S_n)^J$ for the method to represent an alternative to those already known.

The cutting-plane algorithms presented here are part of our theoretical development, and minimally these would have to be supplemented with good heuristic rules to be successful in practice. Furthermore, the "best" ways of using cutting-planes may be within an enumerative framework and with heuristically-found primal solutions (see our discussions in [35, 36]). In this section we have a purely intellectual purpose, and that is to show that the disjunctive cuts do not require the assistance of other devices in order to obtain finite convergence. While some of the algorithms below do have promise and may prove successful when properly implemented, we will not address such practical issues in this section.

First, for a simple case which provides a "subroutine" for the full construction to follow, suppose we wish to solve the following program in linear logical constraints:

$$\begin{aligned} &\min \ cx \\ &\text{subject to } \ Ax \geq b \qquad\qquad (2.2.\text{A}) \\ &\qquad\qquad\quad x \geq 0 \end{aligned}$$

$$\text{and also } \ Q^h x \geq q^h \ \text{ for at least one } h \in H = \{1, \ldots, n\}. \qquad (2.2.\text{B})$$

We assume throughout that $\{x \mid Ax \geq b, x \geq 0\}$ is bounded and non-empty.

The constraints of (2.2.A), (2.2.B) are of the form of those for the exactness result in Proposition (1) of Section 2.1.1, if one uses

$$A^h = \begin{bmatrix} A \\ D^h \end{bmatrix}, \tag{2.2.C}$$

$$b^h = \begin{bmatrix} b \\ d^h \end{bmatrix}, \quad h \in H = \{1, \ldots, t\}. \tag{2.2.D}$$

Hence exactness holds, and all valid cutting-planes are obtained from

$$CT((A^1 x \geq b^1) \vee \cdots \vee (A^t x \geq b^t)).$$

The following strategy suggests itself for (2.2.A), (2.2.B). We can solve (2.2.A) as a linear program without the disjunctive constraints (2.2.B). For all linear programs solved, we assume that an extreme point algorithm is used, i.e., one which provides a solution that is an extreme point whenever the program is consistent and bounded. The Simplex Algorithm is of this type; the subgradient algorithms are not.

If the linear program is inconsistent, we halt: (2.2.A), (2.2.B) is inconsistent. Otherwise, by boundedness, we obtain an optimal extreme point solution x^0. If x^0 satisfies (2.2.B), we are done: it is optimal for (2.2.A), (2.2.B). In what follows, we assume that x^0 does not solve (2.2.B).

We claim that there is at least one facet or singular defining inequality of the set

$$T = \operatorname{clconv}\left(\bigcup_{h \in H} \{x \mid A^h x \geq d^h, x \geq 0\} \right) \tag{2.2.E}$$

which is not satisfied by x^0, i.e., which "cuts off" x^0. Here $A^h x \geq d^h$ includes the constraints appended to $Ax \geq b$ to date, including any previous cuts.

To see the claim, by the boundedness of

$$\{x \mid Ax \geq b, x \geq 0\} \supseteq \bigcup_{h \in H} \{x \mid A^h x \geq d^h, x \geq 0\},$$

the extreme points of the set T are in its generator set

$$\bigcup_{h \in H} \{x \mid A^h x \geq d^h, x \geq 0\},$$

and this set is, in turn, contained in the current linear programming relaxation $A^1 x \geq d^1$, $x \geq 0$. Therefore, if $x^0 \in T$, it would not be an extreme point of the relaxation. This shows $x^0 \notin T$. Therefore, if T is fully-dimensional (as occurs if the constraints (2.2.A), (2.2.B) define a fully dimensional set), there is a facet of T not satisfied by x^0; for T not fully-dimensional, either a facet or a singular inequality of T is not satisfied by x^0.

After the facet or singular inequality is added as a "cutting-plane," the resulting enlarged linear programming relaxation is reoptimized, and the procedure repeats.

We now prove finite convergence of the procedure. Here it is important to note that, after re-optimization, the new set T obtained is the same as the previous one in (2.2.E). This is because the cutting-plane added is satisfied by the set of (2.2.E), hence satisfied by each set $\{x \mid A^w x \ge d^w,\ x \ge 0\}$ for $w \in H$, and so when the matrix $A^w x \ge d^w$ is enlarged by the new inequality, these sets, and therefore their convex span, will not change.

Finite convergence follows simply because the set T of (2.2.E) has only finitely many facets and singular inequalities. After all have been added — and a new one is added each time — certainly (2.2.B) will be satisfied, since an extreme point of T is in one of the sets $\{x \mid A^w x \ge d^w,\ x \ge 0\}$, and therefore satisfies at least one of the conditions of (2.2.B).

The facets of T of (2.2.E) are to be obtained from the co-proposition

$$CT((A^1x \ge d^1) \vee \cdots \vee (A^h x \ge d^h)).$$

This involves finding a suitable face — for T fully dimensional, an extreme ray — of a system like (2.1.3.A), and projecting the (π, π_0) co-ordinates. Not every projection is a facet or singular inequality, but they are among these projections, and only finitely many facets of the desired type exist for (2.1.3.A). Therefore if, at each iteration, we simply add the (π, π_0)-projection of a face for (2.1.3.A), finite convergence is again guaranteed.

In the case that T is fully dimensional, one can set up (2.1.3.A) in terms of the current non-basic variables, turn the desired extreme rays into extreme points by adding $\pi_0 \ge 1$ (since only facets or singular inequalities "cutting-away" x^0 are desired), and determine any extreme point of the resulting system.

Now consider a more complex logical linear program of the form

$$\begin{aligned} \min\ & cx \\ \text{subject to}\ & Ax \ge b \\ & x \ge 0 \\ & cx \in \Sigma \end{aligned} \tag{2.2.F}$$

and also, for every $p \in P = \{1, \ldots, t\}$, we have

$$Q^{p,w}x \ge q^{p,w} \tag{2.2.G}$$

for at least one $w \in H_p = \{1, \ldots, t(p)\}$. In (2.2.F), we require that Σ is a finite set. For instance, (IP) is of this form when it is bounded and $c = (c_1, \ldots, c_n)$ is integral, by taking Σ as the integers, $P = \{1, \ldots, t\}$ with $Q^{p,w}x = q^{p,w}$ as $x_p = w$, w integer, where H_p is sufficiently large so that all possible values of x_p are included. To represent (IP) via (2.2.G) with disjunctive systems of no more than two elements, one may use a number of systems of the form ($x_j \le w$ or $x_j \ge w + 1$).

Of course, by converting the logical constraints (2.2.G) into disjunctive systems $(S_{(\cdot)})$, we can reduce this problem to the one studied in (2.2.A), (2.2.B) above. But the procedure that we now describe uses much smaller disjunctive systems; for (IP), only systems with two conditions need be employed.

The procedure described above for (2.2.A), (2.2.B) will provide our basic step, so again we are using linear optimization to repeatedly solve tighter and tighter linear relaxations. But we shall suppose that this linear optimization is lexicographic with respect to $s(Q^{(1)}), \dots, s(Q^{(\theta)})$, in that order. Here $s(Q^{(p)})$ denotes the sum of the rows of $Q^{(p)}$.

By such a lexicographic method, we mean the following. Reduced cost rows are maintained for the linear forms $s(Q^{(p)})x, p \in P$. First cx is optimized; then if $s(Q^{(1)})x$ can be further decreased without changing the value of cx (i.e., if there are pivots in columns where the criterion function cx has zeroes), these pivots are employed until no more remain; then if $s(Q^{(2)})x$ can be further decreased without changing the value of cx or $s(Q^{(1)})x$, these pivots are employed until no more remain; etc. In this method, $s(Q^{(p)})x$ is given complete priority over $s(Q^{(p+1)})x$, by only using pivots with entering columns that have zeroes in the rows for $s(Q^{(i)})x$, $i < p$.

For an optimal solution x^0 to the current linear programming relaxation, call k the truncation index if: (1) For each $p = 1, \dots, k$ there is $w(p) \in H_p$ such that $Q^{(p)}x^0 = q^{p,w(p)}$; (2) $Q^{(k+1)}x^0 \neq q^{k+1,w}$ for all $w \in H_{k+1}$. I.e., $(k+1)$ is the index of the "first" set of violated constraints. We put $k = 0$ if all logical constraints are violated. If $k = \theta$, we may terminate: x^0 is optimal for (2.2.F), (2.2.G). Assume now that $k < \theta$.

Associated with the truncation index k of x^0 is the vector $(q^{1,w(1)}, \dots, q^{k,w(k)})$ of clause (1), the truncation vector. By definition, the truncation vector is $\emptyset$ if $k = 0$. Here it is important to make the observation that, if the truncation index for the next optimum x^N after re-optimization is $k' < k$, then this truncation vector will never occur again for the same criterion value $z^0 = cx^0$. Indeed, if $z^0 = cx$, since there has occurred a lexicographic decrease in $(cx, s(Q^{(1)})x, \dots, s(Q^{(k')})x)$ with truncation index $k' < k$, for x in all subsequent solutions at least one of the quantities $cx, s(Q^{(1)})x, \dots, s(Q^{(k')})x$ will be less than the corresponding quantity for x^0. But if $Q^{(i)}x = q^{i,w(i)}$ holds, then the value of $s(Q^{(i)})x$ is the sum of elements of $q^{i,w(i)}$. Therefore, for some $i = 1, \dots, k'$, $q^{i,w(i)}$ is not the ith component of any subsequent truncation vector.

When $cx^0 \notin \Sigma$, we add the cut

$$cx \leq \lfloor cx^0 \rfloor_\Sigma \tag{2.2.H}$$

where $\lfloor v \rfloor$ denotes the largest element of Σ that is $\leq v$, for $v \in R$. (If no such element exists, the program is inconsistent.) (2.2.H) certainly causes pivoting to a new point. Otherwise, as in the algorithm for (2.2.A), (2.2.B) we add a facet or singular inequality for the set

$$\operatorname{clconv}\Big(\bigcup_{w \in H_h} \{x \mid A^0x \geq d^0, x \geq 0, Q^{(h)}x = q^{h,w}\}\Big), \quad h = k+1, \tag{2.2.I}$$

by means of the corresponding exact co-proposition. In (2.2.I), $A^0x \geq d^0$, $x \geq 0$ is the current linear programming relaxation, and of course k is the truncation index.

The procedure just described is finite. By the boundedness of $\{x \mid Ax \geq b, x \geq 0\}$ in (2.2.F), (2.2.G), only finitely many cuts of the type (2.2.H) can be added, since Δ is finite. Therefore, to prove finite convergence, it suffices to show that there will not be an infinite sequence of cuts of the type (2.2.H) added all with the same value of $x^0 = cx^0$. Since truncation vectors do not repeat when there is a decrease in the truncation index, and since there are only finitely many truncation vectors, this case simplifies to showing that the truncation index must decrease after finitely many cuts are added.

However, the argument for (2.2.A), (2.2.B) shows that the truncation index cannot remain the same in an infinite, consecutive sequence of cuts. If it decreases, we are done. If it increases, the same analysis repeats for the larger truncation index, and eventually the truncation index cannot increase, since it will reach the upper bound of θ. This completes the proof of finite convergence.

Balas has provided finitely-convergent cutting-plane algorithms, also based on a lexicographic argument, for a class of linear logical programs called "facial" [5]. This class includes the important case of (IP) for bivalent variables.

Acknowledgements

I wish to thank my secretary Jacqueline Cavendish for her Herculean efforts in typing and editing this paper in a brief period of time, an undertaking which seemed at first simply impossible. I am also grateful to Karen Cassel for her generous, voluntary assistance.

On scientific issues, I have been very fortunate in having Egon Balas as a colleague. We have had many conversations on the topics treated here, and I have always benefited from his grasp of the field and his profound insights and extensive knowledge.

I learned several points about the disjunctive methods from Fred Glover, during a visit to Boulder, and Vasek Chvátal's comments on finiteness proofs have been helpful to me.

The intellectual stimulation of many working sessions with Charlie Blair is particularly appreciated, and our collaboration has sharpened my own understanding of several phenomena.

I am very grateful to the sponsors of this Workshop, both the University of Bonn and the IBM Corporation, for their support of this work.

The editors of Discrete Mathematics have been more than helpful in getting this material in final form. I very much appreciate the editorial assistance of Ellis Johnson, and the extraordinary and invaluable care taken by the referee of this very long manuscript.

Lastly, so many thanks to Helen and Avi, for their patient support during July.

References

[1] E. Balas, Intersection cuts — A new type of cutting-planes for integer programming, *Operations Res.* 19 (1971) 19–39.

[2] E. Balas, Integer programming and convex analysis: Intersection cuts from outer polars, *Math. Programming* 2 (1972) 330–382.

[3] E. Balas, On the use of intersection cuts and outer polars in branch and bound, talk given at the MPS conference in Stanford, August 1973.

[4] E. Balas, Intersection cuts from disjunctive constraints, Man. Sci. Res. Rep. No. 330, Carnegie-Mellon University, February 1974.

[5] E. Balas, Disjunctive programming: Facets of the convex hull of feasible points, Man. Sci. Res. Rep. No. 348, Carnegie-Mellon University, July 1974.

[6] E. Balas, Disjunctive programming: Cutting-planes from logical conditions, talk given at SIGMAP-UW Conference, April 1974; in [41, pp. 279–312].

[7] E. Balas and R.G. Jeroslow, Strengthening cuts for mixed integer programs, Man. Sci. Res. Rep. No. 359, Carnegie-Mellon University, February 1975.

[8] C.E. Blair, *Topics in integer programming*, Ph.D. Dissertation, Carnegie-Mellon University, April 1975, 27 pp.

[9] C.E. Blair and R.G. Jeroslow, A note on disjunctive constraints, to appear.

[10] C.-A. Burdet, The algebra and geometry of integer programming cuts: A combined approach, October 1973.

[11] C.-A. Burdet, Polaroids: A new tool in non-convex and in integer programming, *Naval Res. Logistic Quarterly* 20 (1973) 13–24.

[12] C.-A. Burdet, Enumerative inequalities in integer programming, *Math. Programming* 2 (1972) 32–64.

[13] C. Carathéodory, "Über den Variabilitätsbereich der Fourier'schen Konstanten von positiven harmonischen Funktionen," *Rend. Cir. Mat. Palermo* 32 (1911) 193–217.

[14] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Math.* 4 (1973) 305–337.

[15] R.J. Duffin, E.L. Peterson, and C. Zener, *Geometric Programming — Theory and Application* (Wiley, New York, 1967).

[16] M.L. Fisher and J.F. Shapiro, Constructive duality in integer programming, *SIAM J. Appl. Math.* 27 (1974) 31–52.

[17] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming* (Wiley, New York, 1972).

[18] G. Gentzen, "Untersuchungen über das logische Schliessen," *Mathematische Zeitschrift* 39 (1935), pp. 176–210 and pp. 405–431. English translation is #3 in: M.E. Szabo, ed., *The Collected Papers of Gerhard Gentzen* (North-Holland, Amsterdam, 1969) 311 + pp.

[19] A.M. Geoffrion, Lagrangean relaxation for integer programming, *Math. Programming Study* 2 (1974) 82–114.

[20] A.M. Geoffrion and R.E. Marsten, Integer programming algorithms: A framework and state-of-the-art survey, *Management Sci.* 18 (1972) 465–491.

[21] F. Glover, Convexity cuts and cut search, *Operations Res.* 21 (1973) 123–134.

[22] F. Glover, Convexity cuts for multiple-choice problems, MSRS 71-1, University of Colorado, January 1971.

[23] F. Glover, Polyhedral convexity cuts and negative edge extensions, MSRS 73-6, University of Colorado, April 1973.

[24] F. Glover, Polyhedral annexation in mixed integer programming, MSRS 73-9, University of Colorado, August 1973.

[25] F. Glover, On polyhedral annexation and generating the facets of the convex hull of feasible solutions to mixed integer programming problems, MSRS 74-2, University of Colorado, March 1974.

[26] F. Glover and D. Klingman, The generalized lattice-point problem, *Operations Res.* 21 (1973) 135–141.

[27] R.E. Gomory, On the relation between integer and non-integer solutions to linear programs, *Proc. Nat. Acad. Sci.* 53 (1965) 260–265.

[28] R.E. Gomory, An algorithm for integer solutions to linear programs, in: Graves and Wolfe, eds., *Recent Advances in Mathematical Programming* (McGraw-Hill, New York, 1963) pp. 269–302.

[29] R.E. Gomory, An algorithm for the mixed integer problem, RM-2597, RAND Corporation, 1960.

[30] G.A. Gorry and J.F. Shapiro, An adaptive group theoretic algorithm for integer programming problems, *Management Sci.* 17 (1971) 285–306.

[31] P.L. Hammer and S. Rudeanu, Pseudo-Boolean programming, *Operations Res.* 17 (1969) 233–264.

[32] Hoang Tuy, Concave programming under linear constraints, in Russian: *Doklady Akademii Nauk SSSR* (1964) English translation in *Soviet Math.* (1964) 1437–1440.

[33] T.C. Hu, *Integer Programming and Network Flows* (Addison-Wesley, 1969) 452 + pp.

[34] R. Jeroslow, The principles of cutting plane theory: Part I, Carnegie-Mellon University, February 1974.

[35] R. Jeroslow, The principles of cutting-plane theory. Part II: Algebraic methods, disjunctive methods, Man. Sci. Res. Rep. no. 370 (revised), Carnegie-Mellon University, September 1975.

[36] R. Jeroslow, Cutting plane theory: Algebraic methods, February 1976.

[37] R. Jeroslow, Cutting-planes for relaxations of integer programs, Man. Sci. Res. Rep. No. 347, Carnegie-Mellon University, July 1974.

[38] R. Jeroslow, A generalization of a theorem of Chvátal and Gomory, in [41].

[39] E.L. Johnson, Integer programs with continuous variables, July 1974.

[40] E.L. Johnson, The group problem for mixed integer programming, *Math. Programming Study* 2, (1974) 137–179.

[41] O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, *Nonlinear Programming 2* (Academic Press, New York, 1975).

[42] H. Minkowski, *Theorie der Konvexen Körper, insbesondere Begründung ihres Oberflächenbegriffs*, Gesammelte Abhandlungen II, Leipzig, 1911.

[43] G. Owen, Cutting-planes for programs with disjunctive constraints, *J. Optimization Theory and Its Appl.* 11 (1973) 49–55.

[44] D. Prawitz, *Natural Deduction: A Proof-Theoretical Study*, Stockholm Studies in Philosophy 3 (Almqvist and Wiksell, Stockholm, 1965) 105 + pp.

[45] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).

[46] J.F. Shapiro, Generalized Lagrange multipliers in integer programming, *Operations Res.* 19 (1971) 68–76.

[47] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions I* (Springer, Berlin, 1970).

[48] H. Uzawa, A theorem on convex polyhedral cones, in: Arrow, Hurwicz, Uzawa, eds., *Studies in Linear and Nonlinear Programming* (Stanford University Press, Stanford, CA, 1958).

[49] H. Weyl, "Elementare Theorie der Konvexen Polyeder," *Commentarii Mathematici Helvetici* 7 (1935), pp. 290–306; English translation in *Contributions to the Theory of Games*, Kuhn and Tucker, eds., *Ann. Math. Studies* no. 24, Princeton, 1950.

[50] R.D. Young, Hypercylindrically-deduced cuts in zero-one integer programs, *Operations Res.* 19 (1971) 1393–1405.

[51] P. Zwart, Intersection cuts for separable programming, Washington University, St. Louis, January 1972.

Annals of Discrete Mathematics 1 (1977) 331–342

A "PSEUDOPOLYNOMIAL" ALGORITHM FOR SEQUENCING JOBS TO MINIMIZE TOTAL TARDINESS*

Eugene L. LAWLER
Computer Science Division, University of California, Berkeley, CA

Suppose n jobs are to be processed by a single machine. Associated with each job j are a fixed integer processing time p_j, a due date d_j, and a positive weight w_j. The weighted tardiness of job j in a given sequence is $w_j \max\{0, C_j - d_j\}$, where C_j is the completion time of job j. Assume that the weighting of jobs is "agreeable", in the sense that $p_i < p_j$ implies $w_i \geq w_j$. Under these conditions, it is shown that a sequence minimizing total weighted tardiness can be found by a dynamic programming algorithm with worst case running time of $O(n^4P)$ or $O(n^5p_{\max})$, where $P = \sum p_j$ and $p_{\max} = \max\{p_j\}$. The algorithm is "pseudopolynomial", since a true polynomial-bounded algorithm should be polynomial in $\sum \log_2 p_j$.

1. Introduction

Suppose n jobs are to be processed by a single machine. Associated with each job j are a fixed integer processing time p_j, a due date d_j, and a positive weight w_j. The tardiness of job j in a sequence is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j. The problem is to find a sequence which minimizes total weighted tardiness, $\sum w_jT_j$, where the processing of the first job is to begin at time $t = 0$.

Let us assume that the weighting of jobs is *agreeable*, in the sense that $p_i < p_j$ implies $w_i \geq w_j$. Under these conditions, it is shown in this paper that an optimal sequence can be found by a dynamic programming algorithm with worst-case running time of $O(n^4P)$ or $O(n^5p_{\max})$, where $P = \sum p_j$ and $p_{\max} = \max\{p_j\}$.

The proposed algorithm is distinguished from previous algorithms [5, 7, 15] for this problem in that its running time is bounded by a function that is polynomial, rather than exponential, in n. However, the present algorithm does not qualify as a polynomial algorithm in the accepted sense of the term. This is because the running time is not bounded by a polynomial in the number of bits required to specify an instance of the problem in binary encoding. To be polynomial in this sense, the running time should be polynomial in $\sum \log_2 p_j$, rather than P or $p_{\max}$.

Although the proposed algorithm is not polynomial with respect to binary encoding of data, it is polynomial with respect to an encoding in which the p_j values are expressed in unary notation. For this reason, we say that the algorithm is *pseudopolynomial*.

* Research supported by National Science Foundation Grant GJ 43227X.

If the weights of jobs are unrestricted ("disagreeable"), then the weighted tardiness problem is NP-complete, even if all data are encoded in unary notation. (See proof in appendix.) This means that the existence of a pseudopolynomial algorithm is very unlikely. Or more precisely, such an algorithm exists if and only if there are similar algorithms for the traveling salesman problem, the three dimensional assignment problem, the chromatic number problem, and other well-known "hard" problems [6].

It should be mentioned that there is as yet no proof that the agreeably weighted tardiness problem is NP-complete with respect to binary encoding. Hence one may still hope to find a polynomial algorithm. Some unsuccessful attempts are described in the final section of this paper.

There are many closely related types of sequencing problems in which the distinctions between agreeable weighting and unrestricted weighting and between binary encoding and unary encoding are significant. For example, suppose all jobs have the same due date. Then the unrestricted weighted tardiness problem can be solved by a pseudopolynomial algorithm with $O(n^2P)$ complexity [10], whereas the agreeably weighted case yields to an $O(n \log n)$ procedure (SPT order). Or suppose we seek to minimize the weighted *number* of tardy jobs (with respect to arbitrary due dates). The unrestricted problem is NP-complete with respect to binary encoding, but can be solved in $O(nP)$ time [10]. The agreeably weighted case can be solved in $O(n \log n)$ time [9, 11].

2. Theoretical development

Theorem 1. *Let the jobs have arbitrary weights. Let π be any sequence which is optimal with respect to the given due dates $d_1, d_2, \ldots, d_n$, and let C_j be the completion time of job j for this sequence. Let d_j' be chosen such that*

$$\min(d_j, C_j) \le d_j' \le \max(d_j, C_j).$$

Then any sequence π' which is optimal with respect to the due dates $d_1', d_2', \ldots, d_n'$ is also optimal with respect to $d_1, d_2, \ldots, d_n$ (but not conversely).

Proof. Let T denote total weighted tardiness with respect to $d_1, d_2, \ldots, d_n$ and T' denote total weighted tardiness with respect to $d_1', d_2', \ldots, d_n'$. Let π' be any sequence which is optimal with respect to $d_1', d_2', \ldots, d_n'$, and let C_j' be the completion time of job j for this sequence. We have

$$T(\pi) = T'(\pi) - \sum_j A_j, \tag{1.1}$$

$$T(\pi') = T'(\pi') + \sum_j B_j \tag{1.2}$$

where, if $C_j \le d_j$,

$$A_j = 0$$

$$B_j = w_j \max(0, \min(C_j, d_j) - d'_j),$$

and, if $C_j > d_j$,

$$A_j = w_j(d'_j - d_j)$$

$$B_j = w_j \max(0, \min(C_j, d'_j) - d_j)$$

Clearly $A_j \geq B_j$ and $\sum_j A_j \geq \sum_j B_j$. Moreover, $T'(\pi) \leq T'(\pi')$, because π' is assumed to minimize T'. Therefore the right hand side of (1.1) dominates the right hand side of (1.2). It follows that $T(\pi) \geq T(\pi')$ and π' is optimal with respect to $d_1, d_2, \ldots, d_n$. □

Theorem 2. *Suppose the jobs are agreeably weighted. Then there exists an optimal sequence π in which job i precedes job j if $d_i \leq d_j$ and $p_i < p_j$, and in which all on time jobs are in nondecreasing deadline order.*

Proof. Let π be an optimal sequence. Suppose i follows j in π, where $d_i \leq d_j$ and $p_i < p_j$. Then a simple interchange of i and j yields a sequence for which the total weighted tardiness is no greater. (Cf. [13, proof of Theorem 1].) If i follows j, where $d_i \leq d_j$ and i and j are both on time, then moving j to the position immediately following i yields a sequence for which the total weighted tardiness is no greater. Repeated applications of these two rules yields an optimal sequence satisfying the conditions of the theorem. □

In order to simplify exposition somewhat, let us assume for the purposes of the following theorem that all processing times are distinct. If processing times are not distinct, they may be perturbed infinitesimally without upsetting the assumption of agreeable weighting or otherwise changing the problem significantly. Hence there is no loss of generality.

Theorem 3. *Suppose the jobs are agreeably weighted and numbered in nondecreasing due date order, i.e. $d_1 \leq d_2 \leq \cdots \leq d_n$. Let job k be such that $p_k = \max_j \{p_j\}$. Then there is some integer δ, $0 \leq \delta \leq n - k$, such that there exists an optimal sequence π in which k is preceded by all jobs j such that $j \leq k + \delta$, and followed by all jobs j such that $j > k + \delta$.*

Proof. Let C'_k be the latest possible completion time of job k in any sequence which is optimal with respect to due dates $d_1, d_2, \ldots, d_n$. Let π be a sequence which is optimal with respect to the due dates $d_1, d_2, \ldots, d_{k-1}, d'_k = \max(C'_k, d_k), d_{k+1}, \ldots, d_n$, and which satisfies the conditions of Theorem 2 with respect to these due dates. Let C_k be the completion time of job k for π. By Theorem 1, π is optimal with respect to the original due dates. Hence, by assumption, $C_k \leq d_k'$. Job k cannot be preceded in π by any job j such that $d_j > d_k'$, else job j would also be on time, in violation of the conditions of Theorem 2. And job k must be preceded by all jobs j such that $d_j \leq d_k'$. Let δ be chosen to be the largest integer such that $d_{k+\delta} \leq d_k'$ and the theorem is proved. $\square$

3. Dynamic programming solution

Assume the jobs are agreeably weighted and numbered in nondecreasing deadline order. Suppose we wish to find an optimal sequence of jobs $1, 2, \ldots, n$, with processing of the first job to begin at time t. Let k be the job with largest processing time. It follows from Theorem 3 that, for some δ, $0 \leq \delta \leq n - k$, there exists an optimal sequence in the form of,

(i) jobs $1, 2, \ldots, k-1, k+1, \ldots, k+\delta$, in some sequence, starting at time t, followed by

(ii) job k, with completion time $C_k(\delta) = t + \sum_{j \leq k+\delta} p_j$, followed by,

(iii) jobs $k+\delta+1, k+\delta+2, \ldots, n$, in some sequence, starting at time $C_k(\delta)$.

By the well known principle of optimality it follows that the overall sequence is optimal only if the sequences for the subsets of jobs in (i) and (iii) are optimal, for starting times t and $C_k(\delta)$, respectively. This observation suggests a dynamic programming method of solution. For any given subset S of jobs and starting time t, there is a well-defined sequencing problem. An optimal solution for problem S, t can be found recursively from optimal solutions to problems of the form S', t', where S' is a proper subset of S and $t' \geq t$.

The subsets S which enter into the recursion are of a very restricted type. Each subset consists of jobs in an interval $i, i+1, \ldots, j$, with processing times strictly less than some value p_k. Accordingly, denote such a set by

$$S(i, j, k) = \{j' \mid i \leq j' \leq j, p_{j'} < p_k\},$$

and let

$$T(S(i, j, k), t) = \text{the total weighted tardiness for an optimal sequence of the jobs in } S(i, j, k), \text{ starting at time } t.$$

By the application of Theorem 3 and the principle of optimality, we have:

$$\begin{aligned} T(S(i, j, k), t) = \min_{\delta} \{ & T(S(i, k'+\delta, k'), t) + w_{k'} \max(0, C_{k'}(\delta) - d_{k'}) \\ & + T(S(k'+\delta+1, j, k'), C_{k'}(\delta))\} \end{aligned} \tag{3.1}$$

where k' is such that

$$p_{k'} = \max\{p_{j'} \mid j' \in S(i, j, k)\},$$

and

$$C_k(\delta) = t + \sum p_j,$$

where the summation is taken over all jobs $j \in S(i, k+\delta, k')$.

The initial conditions for the equations (3.1) are

$$T(\phi, t) = 0$$

$$T(\{j\}, t) = w_j \max(0, t + p_j - d_j).$$

It is easy to establish an upper bound on the worst-case running time required to compute an optimal sequence for the complete set of n jobs. There are no more than $O(n^3)$ subsets $S(i, j, k)$. (There are no more than n values for each of the indices, i, j, k. Moreover, several distinct choices of the indices may specify the same subset of jobs.) There are surely no more than $P = \sum p_j \leq np_{\max}$ possible values of t. Hence there are no more than $O(n^3P)$ or $O(n^4p_{\max})$ equations (3.1) to be solved. Each equation requires minimization over at most n alternatives and $O(n)$ running time. Therefore the overall running time is bounded by $O(n^4P)$ or $O(n^5p_{\max})$.

At this point we have accomplished the primary objective of this paper, which is to present an algorithm which is polynomial in n. The remaining sections are devoted to a discussion of various computational refinements.

4. Refinements of the algorithm

There are several possible refinements of the basic algorithm that may serve to reduce the running time significantly. However, none of these refinements is sufficient to reduce the theoretical worst-case complexity; some may actually worsen it.

Representation of subsets

It should be noted that $S(i, j, k)$ may denote precisely the same subset of jobs as $S(i', j', k')$ even though $i \neq i'$, $j \neq j'$, $k \neq k'$. The notation used in (3.1) is employed only for convenience in specifying subsets. Obviously, the computation should not be allowed to be redundant.

State generation

Only a very small fraction of the possible subproblems S, t are of significance in a typical calculation. Any practical scheme for implementing the recursion should have two phases. In the first, *subproblem generation* phase, one starts with the problem $S = \{1, 2, \ldots, n\}$, $t = 0$ and successively breaks it down into only those subproblems S, t for which equations (3.1) need to be solved. In the second, *recursion* phase, one solves each of the subproblems generated in the first phase, working in the order opposite to that in which they were generated.

Restriction of δ

It is often not necessary for δ to range over all possible integer values in (3.1). The range of δ can sometimes be considerably restricted by the technique described in the next section, thereby reducing the number of subproblems that need be generated and solved.

Shortcut solutions

There are some "shortcut" methods of solution for the sequencing problem. Whenever one of these shortcut methods is applicable to a subproblem S, t generated in the first phase of the algorithm, it is unnecessary to solve that problem by recursion of the form (3.1) and no further subproblems need be generated from it. A discussion of shortcut solution methods is given in Section 6.

Branch-and-bound

At least in the case of problems of moderate size, there appears to be relatively little duplication of the subproblems produced in the subproblem generation phase of the algorithm. In other words, the recursion tends to be carried out over a set of subproblems related by a tree structure, or something close to it. It follows that there may be some advantage to a branch-and-bound method, based on the structure of equations (3.1). Such a branch-and-bound method might have a very poor theoretical worst-case running time bound, depending on the nature of the bounding calculation and other details of implementation. However, if a depth-first exploration of the search tree is implemented, storage requirements could be very drastically reduced.

It is apparent that the form of recursion (3.1) furnishes a point of departure for the development of many variations of the basic computation.

5. Restriction of δ

The number of distinct values of δ over which minimization must be carried out in equation (3.1) can sometimes be reduced by appropriately invoking Theorems 1 and 2. If this is done in the state generation phase of the algorithm, there may be a considerable reduction in the number of subproblems which must be solved.

Consider a subproblem S, t. Let k be such that

$$p_k = \max_{i \in S} \{p_i\},$$

and assume that $p_k > p_j$ for all $j \in S - \{k\}$. We also assume that the jobs are numbered so that $d_1 \leq d_2 \leq \cdots \leq d_n$. The following algorithm determines distinct values δ_i, $i = 1, 2, \ldots, n - k$, over which it is sufficient to carry out minimization in equation (3.1).

(0) Set $i = 1$.
(1) Set $d'_k = t - \sum_{j \in S'} p_j$ where $S' = \{j \mid d_j \le d_k, j \in S\}$.

Comment. If job k has due date d'_k, then by Theorem 2 there exists an optimal sequence in which the completion time of job k is at least as large as d'_k.

(2) If $d'_k > d_k$ set $d_k = d'_k$ and return to Step 1.

Comment. By Theorem 1, there exists a sequence which is optimal with respect to d'_k which is optimal with respect to d_k.

Let j be the largest index in S such that $d_j \le d_k$. Set $\delta_i = j - k$.
Let $S'' = \{j \mid d_j > d_k, j \in S\}$. If S'' is empty, stop. Otherwise, let j' be such that

$$d_{j'} = \min_{j \in S''} \{d_j\},$$

and set $d_k = d_{j'}$. Set $i = i + 1$ and return to Step 1.

As an example of the application of the above procedure, consider the first test problem given in Appendix A of [1]. All $w_j = 1$. The p_j and d_j values are as follows:

j	1	2	3	4	5	6	7	8
p_j	121	79	147	83	130	102	96	88
d_j	260	266	269	336	337	400	683	719

Note that $k = 3$. Equation (3.1) yields:

$$T(\{1,2,\ldots,8\},0) = \min \begin{Bmatrix} T(S(1,3,3),0) + 78 + T(S(4,8,3),347), \\ T(S(1,4,3),0) + 161 + T(S(5,8,3),430), \\ T(S(1,5,3),0) + 291 + T(S(6,8,3),560), \\ T(S(1,6,3),0) + 393 + T(S(7,8,3),662), \\ T(S(1,7,3),0) + 489 + T(S(8,8,3),758), \\ T(S(1,8,3),0) + 577 + T(\phi,846) \end{Bmatrix}$$

Applying the procedure above, we obtain $\delta_1 = 3$, $\delta_2 = 5$ and the simplified equation:

$$T(\{1,2,\ldots,8\},0) = \min \begin{Bmatrix} T(S(1,6,3),0) + 393 + T(S(7,8,3),662), \\ T(S(1,8,3),0) + 577 + T(\phi,846) \end{Bmatrix} \tag{5.1}$$

6. Shortcut solutions

"Shortcut" solutions are sometimes provided by generalizations of two well-known theorems for the unweighted tardiness problem [3].

Theorem 4. *Let the jobs be given arbitrary weights. Let π be a sequence in which jobs are ordered in nonincreasing order of the ratios w_j/p_j. If all jobs are tardy, then π is optimal.*

Proof. Note that:

$$\sum w_jT_j = \sum w_jC_j - \sum w_j \max\{0, d_j - C_j\} - \sum w_jd_j.$$

It is well-known [14] that π minimizes $\sum w_jC_j$. If all jobs are tardy, then each term in the second summation is zero and that sum is also minimized. □

Note that if jobs are agreeably weighted and processing times are distinct, then w_j/p_j-ratio order is equivalent to shortest processing time order.

Theorem 5. *Let the jobs be given arbitrary weights. Let π be a sequence for which*

$$\max_j \{w_jT_j\}$$

is minimum. If at most one job is tardy, then π is optimal.

Proof. Obvious. □

Note that in the unweighted case, nondecreasing due date order minimizes maximum tardiness. In the case of arbitrary weightings, a minmax optimal order can be constructed by the $O(n^2)$ algorithm given in [8].

The application of these two theorems can be strengthened considerably by applying them to an earlier or a later set of due dates induced by Theorems 1 and 2.

For a problem S, t let the jobs in S be numbered so that $p_1 \geq p_2 \geq \cdots \geq p_n$. New (earlier) deadlines d'_j for the application of Theorem 4 can be induced by the following algorithm.

(0) Set $k = n+1$.
(1) If $k = 1$, stop. Otherwise, set $k = k - 1$.
(2) Set $d'_k = d_k$.
(3) Let $S^{(k)} = \{j \mid j \in S, d_j \geq d'_k, p_j \geq p_k\}$. Set $C_k = t - \sum_{j\in S^{(k)}} p_j$.

Comment. $S^{(k)}$ contains all those jobs which can be assumed to follow k by Theorem 2.

(4) If $C_k < d'_k$, set $d'_k = C_k$ and return to Step 3. Otherwise, return to Step 1.

New (later) due dates d''_j can be induced by the following algorithm.
(0) Set $k = 1$.
(1) If $k = n$, stop. Otherwise, set $k = k + 1$.
(2) Set $d''_k = d_k$.
(3) Let $S^{(k)} = \{j \mid j \in S, d_j \leq d''_k, p_j \leq p_k\}$. Set $C_k = t + p_k - \sum_{j\in S^{(k)}} p_j$.
(4) If $C_k > d''_k$, set $d''_k = C_k$ and return to Step 3. Otherwise, return to Step 1.

By Theorem 2, an optimal solution to the sequencing problem with respect to induced due dates d'_j, $j = 1, 2, \ldots, n$, is optimal with respect to the due dates d_j. Hence Theorems 4 and 5 can be applied with respect to the induced due dates.

As an application of Theorems 4 and 5, let us solve equation (5.2). Consider first the application of Theorem 5 to $S(1,8,3)$, $t = 0$. If the jobs in $S(1,8,3)$ are sequenced in increasing d_j-order, i.e. 1, 2, 4, 5, 6, 7, 8, then jobs 5 and 6 are tardy so Theorem 5 does not apply. However, if induced due dates are computed, it is found that $d'_5 = 515$, with $d'_j = d_j$ for $j \neq 5$. When the jobs are sequenced in increasing d'_j-order, i.e. 1, 2, 4, 6, 5, 7, 8, no jobs are tardy with respect to d'_j. By Theorem 1, the sequence is optimal with respect to the original due dates and $T(S(1,8,3),0) = 178$. Also by Theorem 5, $T(S(1,6,3),0) = 178$. And by Theorem 4, $T(S(7,8,3),662) = 194$. Hence (5.1) becomes:

$$T(\{1,2,\ldots,8\},0) = \min\begin{Bmatrix}178 + 393 + 194, \\ 178 + 577 + 0\end{Bmatrix}$$

$$= 755,$$

as indicated by Baker [1]. An optimal sequence is: 1, 2, 4, 6, 5, 7, 8, 3. Most of the test problems on the same list can be resolved with similar simplicity.

It should be mentioned that even in the case that Theorems 4 and 5 do not yield shortcut solutions, it may be possible to reduce the size of a subproblem with the following observation.

Theorem 6. *Let k be such that $d'_k = \max\{d'_j \mid j \in S\}$, where the d'_j are induced deadlines obtained as above. Let P be the sum of the processing times of jobs in S. If $P + t \geq d'_k$, then*

$$T(S,t) = T(S - \{k\}, t) + w_k \max\{0, P + t - d_k\}.$$

Proof. Cf. [2]. □

7. Possibilities for a polynomial algorithm

As we have commented, the status of the agreeably weighted tardiness problem is unclear. The proposed algorithm is only "pseudopolynomial". However, no problem reduction has been devised to show that the problem is NP-complete, and one may still reasonably suppose that a polynomial algorithm does exist.

There are some possibilities that *do not* seem rewarding in searching for a polynomial algorithm. For a given set S, $T(S,t)$ is a piecewise linear function of t. If $T(S,t)$ were also convex, and all $w_j = 1$, then $T(S,t)$ could be characterized by at most $n + 1$ linear segments, with successive slopes $0, 1, 2, \ldots, n$. The function $T(S,t)$

could then be computed in polynomial time, using equation (3.1). Unfortunately, $T(S, t)$ is *not* convex, as can be shown by simple counterexamples.

If the values of δ for which the minimum is obtained in (3.1) were *monotonically* nondecreasing with t, then this would also suggest a polynomial bounded algorithm. Unfortunately, there are simple counterexamples for this property, as well.

Appendix

The following proof of the unary NP-completeness of the weighted tardiness problem was communicated to the author by M.R. Garey and D.S. Johnson. An alternative proof has been developed by J.K. Lenstra [12].

The so-called 3-partition problem was shown to be unary NP-complete in [4]. This problem is as follows. Given a set of $3n$ integers $a_1, a_2, \ldots, a_{3n}$ between 1 and $B-1$ such that $\sum a_i = nB$, we wish to determine whether there is a partition of the a_i's into n groups of 3, each summing exactly to B.

The corresponding scheduling problem:

"X"-jobs: X_i, $1 \leq i \leq n$.

"A"-jobs: A_i, $1 \leq i \leq 3n$.

Processing times: $p(X_i) = L = (16B^2)\dfrac{n(n+1)}{2} + 1$, $1 \leq i \leq n$,

$p(A_i) = B + a_i$, $1 \leq i \leq 3n$.

Weights: $w(X_i) = W = (L + 4B)(4B)\dfrac{n(n+1)}{2} + 1$, $1 \leq i \leq n$,

$w(A_i) = p(A_i) = B + a_i$, $1 \leq i \leq 3n$.

Due dates: $d(X_i) = iL + (i-1)4B$, $1 \leq i \leq n$,

$d(A_i) = 0$, $1 \leq i \leq 3n$.

Question: Is there a schedule π with total weighted tardiness $T(\pi) \leq W - 1$?

Suppose the desired partition exists. We may assume without loss of generality that the groups are $\{a_{1_j}, a_{2_j}, a_{3_j}\}$, $1 \leq j \leq n$. Consider the following ordering of the jobs:

$$\pi = (X_1, A_{1_1}, A_{2_1}, A_{3_1}, X_2, A_{1_2}, A_{2_2}, A_{3_2}, X_3, \ldots, X_n,$$
$$A_{1_n}, A_{2_n}, A_{3_n}).$$

By assumption $\sum_{k=1}^{3} p(A_{k_j}) = 4B$ for $1 \leq j \leq n$. Thus X_i will finish at time $iL + (i-1)4B = d(X_i)$, $1 \leq i \leq n$, and so none of the X-jobs are tardy. On the other hand, *all* the A-jobs are tardy, with tardinesses equal to their completion times. For each j, $1 \leq j \leq n$, the three jobs A_{3j-2}, A_{3j-1} and A_{3j} all finish by $j(L+4B)$, and their total weight is $4B$. Hence their collective weighted tardiness is at most $j(L+4B)4B$. Hence

$$T(\pi) \leq \sum_{j=1}^{n} j(4B)(L+4B) = \frac{n(n+1)}{2}(4B)(L+4B) = W-1,$$

and π is the desired schedule.

Conversely, suppose that π is such that $T(\pi) \leq W-1$. Clearly no X-job can be tardy, for even a tardiness of 1 would yield $T(\pi) \geq W$. Now define W_i to be the total weight of the A-jobs which follow i X-jobs, with $W_{n+1}=0$ by convention. Then

$$T(\pi) \geq \sum_{i=1}^{n} (W_i - W_{i+1})(iL) = L\sum_{i=1}^{n} W_i.$$

Since all X-jobs meet their due date, we must have $W_i \geq (n-i+1)4B$, $1 \leq i \leq n$. Suppose some $W_i \geq (n-i+1)4B+1$. Then

$$\sum W_i \geq 1 + \sum_{i=1}^{n} (n-i+1)4B = \frac{n(n+1)}{2}(4B)+1.$$

This would imply that

$$T(\pi) \geq L(4B)\left(\frac{n(n+1)}{2}\right) + 16B^2\frac{n(n+1)}{2} + 1 = W,$$

a contradiction.

Thus $W_i = (n-i+1)4B$, $1 \leq i \leq n$. From this we conclude that the set of A-jobs between X_i and X_{i+1} in π has total weight $4B$, $1 \leq i \leq n-1$, and similarly for the set of A-jobs following X_n. Since all A-jobs have $B+1 \leq w(A) \leq 2B-1$, each such set must contain exactly 3 jobs. These n groups of 3 jobs correspond to the desired partition. □

Acknowledgement

The author wishes to acknowledge the helpful comments of Dr. Alexander Rinnooy Kan.

References

[1] K.R. Baker, Introduction to Sequencing and Scheduling, (Wiley, New York, 1974).
[2] S. Elmaghraby, The one-machine sequencing problem with delay costs, J. Industrial Eng. 19 (1968) 187–199.
[3] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, Operations Res. 17 (1969) 701–715.

[4] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comp. 4 (1975) 397–411.

[5] M. Held and R.M. Karp, A dynamic programming approach to sequencing problems, J. Soc. Indust. Appl. Math. 10 (1962) 196–210.

[6] R.M. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45–68.

[7] E.L. Lawler, Sequencing problems with deferral costs, Management Sci. 11 (1964) 280–288.

[8] E.L. Lawler, Optimal sequencing of a single processor subject to precedence constraints, Management Sci. 19 (1973) 544–546.

[9] E.L. Lawler, Sequencing to minimize the weighted number of tardy jobs, to appear in Rev. Française Automat. Informat. Recherche Opérationnelle, Suppl. to 10 (1976) 27–33.

[10] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Sci. 16 (1969) 77–84.

[11] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, Management Sci. 15 (1968) 102–109.

[12] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.

[13] A.H.G. Rinnooy Kan, B.J. Lageweg, J.K. Lenstra, Minimizing total costs in one-machine scheduling, Operations Res. 23 (1975) 908–927.

[14] W.E. Smith, Various optimizers for single-stage production, Naval Res. Logistics Quarterly 3 (1956) 59–66.

[15] V. Srinivasan, A hybrid algorithm for the one-machine sequencing problem to minimize total tardiness, Naval Res. Logistics Quarterly 18 (1971) 317–327.

Annals of Discrete Mathematics 1 (1977) 343-362

COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

J.K. LENSTRA
Mathematisch Centrum, Amsterdam, The Netherlands

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam, The Netherlands

P. BRUCKER
Universität Oldenburg, G.F.R.

We survey and extend the results on the complexity of machine scheduling problems. After a brief review of the central concept of NP-completeness we give a classification of scheduling problems on single, different and identical machines and study the influence of various parameters on their complexity. The problems for which a polynomial bounded algorithm is available are listed and NP-completeness is established for a large number of other machine scheduling problems. We finally discuss some questions that remain unanswered.

1. Introduction

In this paper we study the complexity of machine scheduling problems. Section 2 contains a brief review of recent relevant developments in the theory of computational complexity, centering around the concept of NP-completeness. A classification of machine scheduling problems is given in Section 3. In Section 4 we present the results on the complexity of these problems: a large number of them turns out to be NP-complete. Quite often a minor change in some parameter transforms an NP-complete problem into one for which a polynomial-bounded algorithm is available. Thus, we have obtained a reasonable insight into the location of the borderline between "easy" and "hard" machine scheduling problems, although some questions remain open. They are briefly discussed in Section 5.

2. Complexity theory

Recent developments in the theory of computational complexity as applied to combinatorial problems have aroused the interest of many researchers. The main credit for this must go to S.A. Cook [7] and R.M. Karp [23], who first explored the relation between the classes $\mathscr{P}$ and $\mathscr{NP}$ of (language recognition) problems solvable by *deterministic* and *non-deterministic* Turing machines respectively, in a number of steps *bounded by a polynomial in the length of the input*. With respect to combinatorial optimization, we do not really require mathematically rigorous definitions of these concepts; for our purposes we may safely identify $\mathscr{P}$ with the class of problems for which a *polynomial-bounded*, *good* [8] or *efficient algorithm* exists, whereas all problems in $\mathscr{NP}$ can be solved by *polynomial-depth backtrack search*.

In this context, all problems are stated in terms of *recognition* problems which require a yes/no answer. In order to deal with the complexity of a combinatorial *minimization* problem, we transform it into the problem of determining the existence of a solution with value at most equal to y, for some *threshold* y.

It is clear that $\mathscr{P} \subset \mathscr{NP}$, and the question arises if this inclusion is a proper one or if, on the contrary, $\mathscr{P} = \mathscr{NP}$. Although this is still an open problem, the equality of $\mathscr{P}$ and $\mathscr{NP}$ is considered to be very unlikely and most bets (e.g., in [28]) have been going in the other direction. To examine the consequences of an affirmative answer to the $\mathscr{P} = \mathscr{NP}$ question, we introduce the following concepts.

Problem P' is *reducible* to problem P (notation: $P' \propto P$) if for any instance of P' an instance of P can be constructed in polynomial-bounded time such that solving the instance of P will solve the instance of P' as well.

P' and P are *equivalent* if $P' \propto P$ and $P \propto P'$.

P is *NP-complete* [28] if $P \in \mathscr{NP}$ and $P' \propto P$ for every $P' \in \mathscr{NP}$. Informally, the reducibility of P' to P implies that P' can be considered as a special case of P; the NP-completeness of P indicates that P is, in a sense, the most difficult problem in $\mathscr{NP}$.

In a remarkable paper [7], NP-completeness was established with respect to the so-called Satisfiability problem. This problem can be formulated as follows.

Given clauses $C_1, \ldots, C_u$, each being a disjunction of literals from the set $X = \{x_1, \ldots, x_t, \bar{x}_1, \ldots, \bar{x}_t\}$, is the conjunction of the clauses satisfiable, i.e., does there exist a subset $S \subset X$ such that S does not contain a complementary pair of literals $(x_i, \bar{x}_i)$, and $S \cap C_j \neq \emptyset$ for $j = 1, \ldots, u$?

Cook proved this result by specifying a polynomial-bounded "master reduction" which, given $P \in \mathscr{NP}$, constructs for any instance of P an equivalent boolean expression in conjunctive normal form. By means of this reduction, a polynomial-bounded algorithm for the Satisfiability problem could be used to construct a polynomial-bounded algorithm for any problem in $\mathscr{NP}$. It follows that

$$\mathscr{P} = \mathscr{NP} \textit{ if and only if } \text{Satisfiability} \in \mathscr{P}.$$

The same argument applies if we replace Satisfiability by any NP-complete problem. A large number of such problems has been identified by Karp [25; 26] and others (e.g., [17]); Theorem 1 mentions some of them. Since they are all notorious combinatorial problems for which typically no good algorithms have been found so far, these results afford strong circumstantial evidence that $\mathscr{P}$ is a proper subset of $\mathscr{NP}$.

Theorem 1. *The following problems are NP-complete.*

(a) Clique. *Given an undirected graph $G = (V, E)$ and an integer k, does G have a clique (i.e., a complete subgraph) on k vertices?*

(b) Linear arrangement. *Given an undirected graph $G = (V, E)$ and an integer k, does there exist a one-to-one function $\pi : V \to \{1, \ldots, |V|\}$ such that $\sum_{\{i,j\}\in E} |\pi(i) - \pi(j)| \leq k$?*

(c) Directed hamiltonian circuit. *Given a directed graph $G = (V, A)$, does G have a hamiltonian circuit (i.e., a directed cycle passing through each vertex exactly once)?*

(d) Directed hamiltonian path. *Given a directed graph $G' = (V', A')$, does G' have a hamiltonian path (i.e., a directed path passing through each vertex exactly once)?*

(e) Partition. *Given positive integers $a_1, \ldots, a_t$, does there exist a subset $S \subset T = \{1, \ldots, t\}$ such that $\sum_{i\in S} a_i = \sum_{i\in T-S} a_i$?*

(f) Knapsack. *Given positive integers $a_1, \ldots, a_t, b$, does there exist a subset $S \subset T = \{1, \ldots, t\}$ such that $\sum_{i\in S} a_i = b$?*

(g) 3-Partition. *Given positive integers $a_1, \ldots, a_{3t}, b$, does there exist a partition $(T_1, \ldots, T_t)$ of $T = \{1, \ldots, 3t\}$ such that $|T_j| = 3$ and $\sum_{i\in T_j} a_i = b$ for $j = 1, \ldots, t$?*

Proof. (a) See [7; 25].

(b) See [17].

(c, e, f) See [25].

(d) NP-completeness of this problem is implied by two observations:

(A) Directed hamiltonian path $\in \mathcal{NP}$,

(B) $P \propto$ Directed hamiltonian path for some NP-complete problem P.

(A) is trivially true, and (B) is proved by the following reduction.

Directed hamiltonian circuit $\propto$ Directed hamiltonian path.

Given $G = (V, A)$, we choose $v' \in V$ and construct $G' = (V', A')$ with

$$V' = V \cup \{v''\},$$

$$A' = \{(v, w) \mid (v, w) \in A, w \neq v'\} \cup \{(v, v'') \mid (v, v') \in A\}.$$

G has a hamiltonian circuit if and only if G' has a hamiltonian path.

(g) See [12]. □

Karp's work has led to a large amount of research on the location of the borderline separating the "easy" problems (in $\mathcal{P}$) from the "hard" (NP-complete) ones. It turns out that a minor change in a problem parameter (notably — for some as yet mystical reason — an increase from two to three) often transforms an easy problem into a hard one. Not only does knowledge of the borderline lead to fresh insights as to what characteristics of a problem determine its complexity, but there are also important consequences with respect to the solution of these problems. Establishing NP-completeness of a problem can be interpreted as a formal justification to use enumerative methods such as branch-and-bound, since no substantially better method is likely to exist. Embarrassing incidents such as the presentation in a standard text-book of an enumerative approach to the undirected Chinese postman problem, for which a good algorithm had already been developed in [9], will then occur less readily.

The class of machine scheduling problems seems an especially attractive object for this type of research, since their structure is relatively simple and there exist standard problem parameters that have demonstrated their usefulness in previous research.

Before describing this class of problems, let us emphasize that membership of $\mathcal{P}$ versus NP-completeness only yields a very coarse measure of complexity. On one hand, the question has been raised whether polynomial-bounded algorithms are really good [2]. On the other hand, there are significant differences in complexity within the class of NP-complete problems.

One possible refinement of the complexity measure may be introduced at this stage. It is based on the way in which the problem data are encoded. Taking the Knapsack and 3-Partition problems as examples and defining $a_* = \max_{j\in T}\{a_j\}$, we observe that the length of the input is $O(t \log a_*)$ in the standard *binary* encoding, and $O(ta_*)$ if a *unary* encoding is allowed. 3-Partition has been proved NP-complete even with respect to a unary encoding [12]. Knapsack is NP-complete with respect to a binary encoding [25], but solution by dynamic programming requires $O(tb)$ steps and thus yields a polynomial-bounded algorithm with respect to a unary encoding; similar situations exist for several machine scheduling problems. Such "pseudopolynomial" algorithms [35] need not necessarily be "good" in the practical sense of the word, but it may pay none the less to distinguish between complexity results with respect to unary and binary encodings (cf. [16]). *Unary NP-completeness* or *binary membership of* $\mathcal{P}$ would then be the strongest possible result, and it is quite feasible for a problem to be *binary NP-complete* and to allow a *unary polynomial-bounded* solution. The results in this paper hold with respect to the standard binary encoding; some consequences of using a unary encoding will be pointed out as well.

3. Classification

Machine scheduling problems can be verbally formulated as follows [6; 45]:

A *job* J_i $(i = 1,\ldots,n)$ consists of a sequence of *operations*, each of which corresponds to the uninterrupted processing of J_i on some *machine* M_k $(k = 1,\ldots,m)$ during a given period of time. Each machine can handle at most one job at a time. What is according to some overall *criterion* the optimal *processing order* on each machine?

The following data can be specified for each J_i:

a *number of operations* n_i;

a *machine order* ν_i, i.e. an ordered n_i-tuple of machines;

a *processing time* p_{ik} of its kth operation, $k = 1, \ldots, n_i$ (if $n_i = 1$ for all J_i, we shall usually write p_i instead of p_{i1});

a *weight* w_i;

a *release date* or *ready time* r_i, i.e. its earliest possible starting time (unless stated otherwise, we assume that $r_i = 0$ for all J_i);

a *due date* or *deadline* d_i;

a *cost function* $f_i : \mathbf{N} \to \mathbf{R}$, indicating the costs incurred as a nondecreasing function of the completion time of J_i.

We assume that all data (except ν_i and f_i) are nonnegative integers. Given a processing order on each M_k, we can compute for each J_i:

the *starting time* S_i;

the *completion time* C_i;

the *lateness* $L_i = C_i - d_i$;

the *tardiness* $T_i = \max\{0, C_i - d_i\}$;

$U_i =$ *if* $C_i \leq d_i$ *then* 0 *else* 1.

Machine scheduling problems are traditionally classified by means of four parameters n, m, l, κ. The first two parameters are integer variables, denoting the numbers of jobs and machines respectively; the cases in which m is constant and equal to 1, 2, or 3 will be studied separately. If $m > 1$, the third parameter takes on one of the following values:

$l = F$ in a *flow-shop* where $n_i = m$ and $\nu_i = (M_1, \ldots, M_m)$ for each J_i;

$l = P$ in a *permutation flow-shop*, i.e. a flow-shop where passing is not permitted so that each machine has to process the jobs in the same order;

$l = G$ in a (*general*) *job-shop* where n_i and ν_i may vary per job;

$l = I$ in a *parallel-shop* where each job has to be processed on just one of m *identical* machines, i.e. $n_i = 1$ for all J_i and the ν_i are not defined.

Extensions to the more general situation where *several groups of parallel* (*possibly non-identical*) *machines* are available will not be considered.

The fourth parameter indicates the optimality criterion. We will only deal with *regular* criteria, i.e., monotone functions κ of the completion times $C_1, \ldots, C_n$ such that

$$C_i \leq C_i' \text{ for all } i \implies \kappa(C_1, \ldots, C_n) \leq \kappa(C_1', \ldots, C_n').$$

These functions are usually of one of the following types:

$\kappa = f_{\max} = \max_i \{f_i(C_i)\}$;

$\kappa = \sum f_i = \sum_{i=1}^n f_i(C_i)$.

The following specific criteria have frequently been chosen to be minimized:

$\kappa = C_{\max} = \max_i \{C_i\}$;

$\kappa = \sum w_i C_i = \sum_{i=1}^n w_i C_i$;

$\kappa = L_{\max} = \max_i \{L_i\}$;

$\kappa = \sum w_i T_i = \sum_{i=1}^n w_i T_i$;

$\kappa = \sum w_i U_i = \sum_{i=1}^n w_i U_i$.

We refer to [45] for relations between these and other objective functions.

Some relevant problem variations are characterized by the presence of one or more elements from a parameter set β, such as

prec (precedence constraints between the jobs, where "J_i precedes J_j" (notation: $J_i < J_j$) implies $C_i \le S_j$);

tree (precedence constraints between the jobs such that the associated precedence graph can be given as a *branching*, i.e. a set of directed trees with either indegree or outdegree at most one for all vertices);

$r_i \ge 0$ (possibly non-equal release dates for the jobs);

$C_i \le d_i$ (all jobs have to meet their deadlines; in this case we assume that $\kappa \in \{C_{\max}, \sum w_i C_i\}$);

no wait (no waiting time for the jobs between their starting and completion times; hence, $C_i = S_i + \sum_k p_{ik}$ for each J_i);

$n_i \le n_+$ (a constant upper bound on the number of operations per job);

$p_{ik} \le p_+$ (a constant upper bound on the processing times);

$p_{ik} = 1$ (unit processing times);

$w_i = 1$ (equality of the weights; we indicate this case also by writing $\sum C_i$, $\sum T_i$, $\sum U_i$).

In view of the above discussion, we can use the notation $n \mid m \mid \beta \mid \kappa$ to indicate specific machine scheduling problems.

4. Complexity of machine scheduling problems

All machine scheduling problems of the type defined in Section 3 can be solved by polynomial-depth backtrack search and thus are members of $\mathcal{NP}$. The results on their complexity are summarized in Table 1.

The problems which are marked by an asterisk (*) are solvable in polynomial-bounded time. In Table 2 we provide for most of these problems references where the algorithm in question can be found; we give also the order of the number of steps in the currently best implementations. The problems marked by a note of exclamation (!) are NP-complete. The reductions to these problems are listed in Table 3. Question-marks (?) indicate open problems. We will return to them in Section 5 to motivate our typographical suggestion that these problems are likely to be NP-complete.

Table 1 contains the "hardest" problems that are known to be in $\mathcal{P}$ and the "easiest" ones that have been proved to be NP-complete. In this respect, Table 1 indicates to the best of our knowledge the location of the borderline between easy and hard machine scheduling problems.

Before proving the theorems mentioned in Table 3, we will give a simple example of the interaction between tables and theorems by examining the status of the general job-shop problem, indicated by $n \mid m \mid G \mid C_{\max}$.

Table 1. Complexity of machine scheduling problems

n jobs	1 machine	2 machines	m machines
C_{max}	* prec, $r_j \geq 0$	* F * F, no wait ! F, tree ! F, $r_j \geq 0$	! $m = 3$: F ? $m = 3$: F, no wait ! F, no wait
		* G, $n_i \leq 2$! G, $n_i \leq 3$	* $n = 2$: G ! $m = 3$: G, $n_i \leq 2$
		! P * P, prec, $r_j \geq 0$, $C_j \leq d_j$, $p_j = 1$! P, prec, $p_j \leq 2$	* P, tree, $p_j = 1$? $m = 3$: P, prec, $p_j = 1$! P, prec, $p_j = 1$
$\sum w_j C_j$	* tree ! prec, $p_j = 1$! prec, $w_j = 1$! $r_j \geq 0$, $w_j = 1$ * $C_j \leq d_j$, $w_j = 1$! $C_j \leq d_j$	! F, $w_j = 1$? F, no wait, $w_j = 1$! P * P, prec, $p_j = 1$, $w_j = 1$! P, prec, $p_j \leq 2$, $w_j = 1$	? F, no wait, $w_j = 1$ * P, $r_j \geq 0$, $p_j = 1$ * P, $w_j = 1$! P, prec, $p_j = 1$, $w_j = 1$
L_{max}	* prec * prec, $r_j \geq 0$, $p_j = 1$! $r_j \geq 0$	! F ! P	
$\sum w_j T_j$	* $r_j \geq 0$, $p_j = 1$? $w_j = 1$! ! prec, $p_j = 1$, $w_j = 1$! $r_j \geq 0$, $w_j = 1$	! F, $w_j = 1$! P, $w_j = 1$	
$\sum w_j U_j$	* $r_j \geq 0$, $p_j = 1$ * $w_j = 1$! ! prec, $p_j = 1$, $w_j = 1$! $r_j \geq 0$, $w_j = 1$	! F, $w_j = 1$! P, $w_j = 1$	

*: problem in $\mathscr{P}$; see Table 2.
?: open problem; see Section 5.
!: NP-complete problem; see Table 3.

Table 2. References to polynomial-bounded algorithms

Problem	References	Order
$n\|1\|prec, r_i \geq 0\|C_{max}$	—	$O(n^2)$
$n\|1\|tree\|\sum w_iC_i$	[20; 1; 46][a]	$O(n \log n)$
$n\|1\|C_i \leq d_i\|\sum C_i$	[47]	$O(n \log n)$
$n\|1\|prec\|L_{max}$	[33]	$O(n^2)$
$n\|1\|prec, r_i \geq 0, p_i = 1\|L_{max}$	[30]	$O(n^2)$
$n\|1\|r_i \geq 0, p_i = 1\|\sum w_iT_i$	[32]	$O(n^3)$
$n\|1\|r_i \geq 0, p_i = 1\|\sum w_iU_i$	[29]	$O(n^2)$
$n\|1\|\sum U_i$	[41][b]	$O(n \log n)$
$n\|2\|F\|C_{max}$	[24]	$O(n \log n)$
$n\|2\|F, no\ wait\|C_{max}$	[18, 44]	$O(n^2)$
$n\|2\|G, m_i \leq 2\|C_{max}$	[23]	$O(n \log n)$
$n\|2\|I, prec, r_i \geq 0, C_i \leq d_i, p_i = 1\|C_{max}$	[14]	$O(n^2)$
$n\|2\|I, prec, p_i = 1\|\sum C_i$	[5; 11]	$O(n^2)$
$2\|m\|G\|C_{max}$	[46; 15]	$O(m^2)$
$n\|m\|I, tree, p_i = 1\|C_{max}$	[22]	$O(n)$
$n\|m\|I, r_i \geq 0, p_i = 1\|\sum w_iC_i$	[32]	$O(n^3)$
$n\|m\|I\|\sum C_i$	[6][d]	$O(n \log n)$

[a] An $O(n \log n)$ algorithm for the more general case of *series parallel precedence constraints* is given in [36].

[b] An $O(n \log n)$ algorithm for the more general case of *agreeable weights* (i.e. $p_i < p_j \Rightarrow w_i \geq w_j$) is given in [34].

[c] $O(n^2)$ and $O(n^3)$ algorithms for the $n|2|I, prec, p_i = 1|C_{max}$ problem are given in [10] and [5] respectively; see also [13].

[d] Polynomial-bounded algorithms for the more general case of *parallel non-identical machines* are given in [21, 4].

In Table 1, we see that the $n|2|G, m_i \leq 2|C_{max}$ problem is a member of $\mathscr{P}$ and that two minor extensions of this problem, $n|2|G, m_i \leq 3|C_{max}$ and $n|3|G, m_i \leq 2|C_{max}$, are NP-complete. By Theorem 2(c, h), these problems are special cases of the general job-shop problem, which is thus shown to be NP-complete by Theorem 2(b). Table 2 refers to an $O(n \log n)$ algorithm [23] for the $n|2|G, m_i \leq 2|C_{max}$ problem. Table 3 tells us that reductions of Knapsack to both NP-complete problems are presented in Theorem 4(a, b); the NP-completeness of Knapsack has been mentioned in Theorem 1(f).

Theorem 2 gives some elementary results on reducibility among machine scheduling problems. It can be used to establish either membership of $\mathscr{P}$ or NP-completeness for problems that are, roughly speaking, either not harder than the polynomially solvable ones or not easier than the NP-complete ones in Table 1.

Theorem 3. (a) *If* $n'|m'|l', \lambda'|\kappa' \propto n|m|l, \lambda|\kappa$ *and* $n|m|l, \lambda|\kappa \in \mathscr{P}$, *then* $n'|m'|l', \lambda'|\kappa' \in \mathscr{P}$.

Table 5. Reductions to NP-complete machine scheduling problems

Reduction	References
Linear arrangement $\propto n\,|\,1\,|\,prec, p_i = 1\,|\,\sum w_iC_i$	[36; 38; 40]
Linear arrangement $\propto n\,|\,1\,|\,prec\,|\,\sum C_i$	[36; 38; 40]
3-Partition $\propto n\,|\,1\,|\,r_i \geq 0\,|\,\sum C_i$	h.l. Theorem 5
Knapsack $\propto n\,|\,1\,|\,d_i = d\,|\,\sum w_iU_i$	h.l. Theorem 4(j)
Knapsack $\propto n\,|\,1\,|\,r_i \geq 0\,|\,L_{max}$	h.l. Theorem 4(c)
Knapsack $\propto n\,|\,1\,|\,\sum w_iT_i$	h.l. Theorem 4(i)
Clique $\propto n\,|\,1\,|\,prec, p_i = 1\,|\,\sum T_i$	[38; 40]
$n\,|\,1\,|\,r_i \geq 0\,|\,L_{max} \propto n\,|\,1\,|\,r_i \geq 0\,|\,\sum T_i$	h.l. Theorem 2(j)
Knapsack $\propto n\,|\,1\,|\,\sum w_iU_i$	[35]; h.l. Theorem 4(h)
Clique $\propto n\,|\,1\,|\,prec, p_i = 1\,|\,\sum U_i$	[12; 38; 40]
$n\,|\,1\,|\,r_i \geq 0\,|\,L_{max} \propto n\,|\,1\,|\,r_i \geq 0\,|\,\sum U_i$	h.l. Theorem 2(j)
Knapsack $\propto n\,|\,2\,|\,F, tree\,|\,C_{max}$	h.l. Theorem 4(f)
Knapsack $\propto n\,|\,2\,|\,F, r_i \geq 0\,|\,C_{max}$	h.l. Theorem 4(d)
Knapsack $\propto n\,|\,2\,|\,G, n_i \leq 3\,|\,C_{max}$	h.l. Theorem 4(a)
Partition $\propto n\,|\,2\,|\,I\,|\,C_{max}$	h.l. Theorem 2(a); cf. [4]
Clique $\propto n\,|\,2\,|\,I, prec, p_i \leq 2\,|\,C_{max}$	[40]; cf. [49]
3-Partition $\propto n\,|\,2\,|\,F\,|\,\sum C_i$	[16]
Partition $\propto n\,|\,2\,|\,I\,|\,\sum w_iC_i$	h.l. Theorem 2(b); cf. [4]
$n\,|\,2\,|\,I, prec, p_i \leq 2\,|\,C_{max} \propto n\,|\,2\,|\,I, prec, p_i \leq 2\,|\,\sum C_i$	h.l. Theorem 2(l); cf. [40]
Knapsack $\propto n\,|\,2\,|\,F\,|\,L_{max}$	h.l. Theorem 4(e)
$n\,|\,2\,|\,I\,|\,C_{max} \propto n\,|\,2\,|\,I\,|\,L_{max}$	h.l. Theorem 2(i)
$n\,|\,2\,|\,F\,|\,L_{max} \propto n\,|\,2\,|\,F\,|\,\sum T_i$	h.l. Theorem 2(j)
$n\,|\,2\,|\,I\,|\,L_{max} \propto n\,|\,2\,|\,I\,|\,\sum T_i$	h.l. Theorem 2(j)
$n\,|\,2\,|\,F\,|\,L_{max} \propto n\,|\,2\,|\,F\,|\,\sum U_i$	h.l. Theorem 2(j)
$n\,|\,2\,|\,I\,|\,L_{max} \propto n\,|\,2\,|\,I\,|\,\sum U_i$	h.l. Theorem 2(j)
Knapsack $\propto n\,|\,3\,|\,F\,|\,C_{max}$	h.l. Theorem 4(g)
Directed hamiltonian path $\propto n\,|\,m\,|\,F, no\ wait\,|\,C_{max}$	h.l. Theorem 6(a)
Knapsack $\propto n\,|\,3\,|\,G, n_i \leq 2\,|\,C_{max}$	h.l. Theorem 4(b)
Clique $\propto n\,|\,m\,|\,I, prec, p_i = 1\,|\,C_{max}$	[40]; cf. [49]
Directed hamiltonian path $\propto n\,|\,m\,|\,F, no\ wait\,|\,\sum C_i$	h.l. Theorem 6(b)
$n\,|\,m\,|\,I, prec, p_i = 1\,|\,C_{max} \propto n\,|\,m\,|\,I, prec, p_i = 1\,|\,\sum C_i$	h.l. Theorem 2(l); cf. [40]

(b) *If* $n'\,|\,m'\,|\,\ell', \lambda'\,|\,\kappa' \propto n\,|\,m\,|\,\ell, \lambda\,|\,\kappa$ *and* $n'\,|\,m'\,|\,\ell', \lambda'\,|\,\kappa'$ *is* **NP-complete**, *then* $n\,|\,m\,|\,\ell, \lambda\,|\,\kappa$ *is* NP-*complete.*

(c) $n\,|\,m'\,|\,\ell, \lambda\,|\,\kappa \propto n\,|\,m\,|\,\ell, \lambda\,|\,\kappa$ *if* $m' \leq m$ *or if* m' *is constant and* m *is variable.*

(d) $n\,|\,2\,|\,F\,|\,\kappa$ *and* $n\,|\,2\,|\,P\,|\,\kappa$ *are equivalent.*

(e) $n\,|\,3\,|\,F\,|\,C_{max}$ *and* $n\,|\,3\,|\,P\,|\,C_{max}$ *are equivalent.*

(f) $n\,|\,m\,|\,F, \lambda\,|\,\kappa \propto n\,|\,m\,|\,G, \lambda\,|\,\kappa$.

(g) $n\,|\,m\,|\,\ell, \lambda\,|\,\kappa \propto n\,|\,m\,|\,\ell, \lambda \cup \lambda'\,|\,\kappa$ *if* $\lambda' \subset \{prec, tree, r_i \geq 0, C_i \leq d_i\}$.

(h) $n\,|\,m\,|\,\ell, \lambda \cup \lambda'\,|\,\kappa \propto n\,|\,m\,|\,\ell, \lambda\,|\,\kappa$ *if* $\lambda' \subset \{n_i \leq n_+, p_{ij} \leq p_+, p_{ij} = 1, w_i = 1\}$.

(i) $n\,|\,m\,|\,\ell, \lambda\,|\,C_{max} \propto n\,|\,m\,|\,\ell, \lambda\,|\,L_{max}$.

(j) $n\,|\,m\,|\,\ell, \lambda\,|\,L_{max} \propto n\,|\,m\,|\,\ell, \lambda\,|\,\kappa$ *if* $\kappa \in \{\sum T_i, \sum U_i\}$.

(k) $n\,|\,m\,|\,\ell, \lambda\,|\,\sum w_iC_i \propto n\,|\,m\,|\,\ell, \lambda\,|\,\sum w_iT_i$.

(l) $n\,|\,m\,|\,I, prec, p_i \leq p_+\,|\,C_{max} \propto n\,|\,m\,|\,I, prec, p_i \leq p_+\,|\,\sum C_i$.

Proof. Let P' and P denote the problems on the left-hand side and right-hand side respectively.

(a, b) Clear from the definition of reducibility.

(c) Trivial.

(d, e) P' has an optimal solution with the same processing order on each machine [6; 45].

(f, g, h) In each case P' obviously is a special case of P.

(i) Given any instance of P' and a threshold value y', we construct a corresponding instance of P by defining $d_i = y'$ $(i = 1, \ldots, n)$. P' has a solution with value $\leq y'$ if and only if P has a solution with value ≤ 0.

(j) Given any instance of P' with due dates d_i' $(i = 1, \ldots, n)$ and a threshold value y', we construct a corresponding instance of P by defining $d_i = d_i' + y'$ $(i = 1, \ldots, n)$. P' has a solution with value $\leq y'$ if and only if P has a solution with value ≤ 0.

(k) Take $d_i = 0$ $(i = 1, \ldots, n)$ in P.

(l) Given any instance of P' and a y', $0 \leq y' \leq n' p_{\max}$, we construct a corresponding instance of P by defining

$$n'' = (n' - 1)\, y',$$
$$n = n' + n'',$$
$$y = ny' + \tfrac{1}{2} n''(n'' + 1),$$

and adding n'' jobs $J_{n'+j}$ $(j = 1, \ldots, n'')$ to P' with

$$p_{n'+j} = 1,$$

$$J_i < J_{n'+j} \quad (i = 1, \ldots, n';\ j = 1, \ldots, n'').$$

Now P' has a solution with value $\leq y'$ if and only if P has a solution with value $\leq y$:

$$C_{\max} \leq y' \implies \sum C_j \leq n'y' + \textstyle\sum_{j=1}^{n''} (y' + j) = y;$$

$$C_{\max} > y' \implies \sum C_j > y' + \textstyle\sum_{j=1}^{n''} (y' + 1 + j) = y. \quad \square$$

Remark. The proof of Theorem 2(c) involves processing times equal to 0, implying that the operations in question require an infinitesimally small amount of time. Whenever these reductions are applied, the processing times can be transformed into strictly positive integers by sufficiently (but polynomially) inflating the problem data. Examples of such constructions can be found in the proofs of Theorem 4(c, d, e, f).

In Theorems 3 to 6 we present a large number of reductions of the form $P \propto \alpha\,|\,\beta\,|\,\gamma$ by specifying $\alpha\,|\,\beta\,|\,\gamma$ and some y such that P has a solution if and only if $\alpha\,|\,\beta\,|\,\gamma$ has a solution with value $\gamma \leq y$. This equivalence is proved for some principal reductions; in other cases, it is trivial or clear from the analogy to a reduction given previously. The NP-completeness of $\alpha\,|\,\beta\,|\,\gamma$ then follows from the NP-completeness of P as established in Theorem 1.

First, we briefly deal with the problems on *identical* machines. Theorem 3 presents two reductions which are simplified versions of the reductions given in [4].

Theorem 3. Partition *is reducible to the following problems*:
(a) $n \,|\, 2 \,|\, I \,|\, C_{\max}$;
(b) $n \,|\, 2 \,|\, I \,|\, \sum w_i C_i$.

Proof. *Define* $A = \sum_{i \in T} a_i$.
(a) Partition $\propto$ $n \,|\, 2 \,|\, I \,|\, C_{\max}$:

$$n = t;$$
$$p_i = a_i \quad (i \in T);$$
$$y = \tfrac{1}{2}A.$$

(b) Partition $\propto$ $n \,|\, 2 \,|\, I \,|\, \sum w_i C_i$:

$$n = t;$$
$$p_i = w_i = a_i \quad (i \in T);$$
$$y = \sum_{1 \le i \le j \le t} a_i a_j - \tfrac{1}{4}A^2.$$

Suppose that $\{J_i \mid i \in S\}$ is assigned to M_1 and $\{J_i \mid i \in T - S\}$ to M_2; let $c = \sum_{i \in S} a_i - \frac{1}{2}A$. Since $p_i = w_i$ for all i, the value of $\sum w_i C_i$ is not influenced by the ordering of the jobs on the machines and only depends on the choice of S [6]:

$$\sum w_i C_i = \kappa(S).$$

It is easily seen (cf. Fig. 1) that

$$\kappa(S) = \kappa(T) - \Big(\sum_{i \in S} a_i\Big)\Big(\sum_{i \in T-S} a_i\Big)$$
$$= \sum_{1 \le i \le j \le t} a_i a_j - (\tfrac{1}{2}A + c)(\tfrac{1}{2}A - c) = y + c^2,$$

and it follows that Partition has a solution if and only if this $n \,|\, 2 \,|\, I \,|\, \sum w_i C_i$ problem has a solution with value $\le y$. $\square$

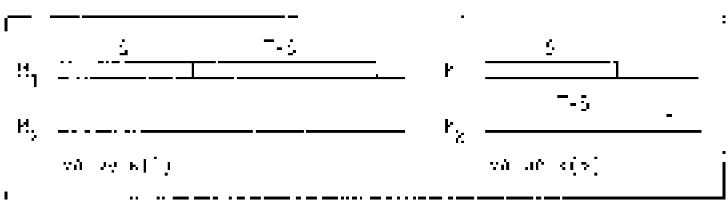


Fig. 1

Most of our results on *different* machines involve the Knapsack problem, as demonstrated by Theorem 4.

Theorem 4. Knapsack *is reducible to the following problems*:
(a) $n \mid 2 \mid G, n_i \leq 3 \mid C_{\max}$;
(b) $n \mid 3 \mid G, n_i \leq 2 \mid C_{\max}$;
(c) $n \mid 3 \mid F \mid C_{\max}$;
(d) $n \mid 2 \mid F, r_i \geq 0 \mid C_{\max}$;
(e) $n \mid 2 \mid F \mid L_{\max}$;
(f) $n \mid 2 \mid F, tree \mid C_{\max}$;
(g) $n \mid 1 \mid r_i \geq 0 \mid L_{\max}$;
(h) $n \mid 1 \| \sum w_i U_i$;
(i) $n \mid 1 \| \sum w_i T_i$;
(j) $n \mid 1 \mid C_i \leq d_i \mid \sum w_i C_i$.

Proof. Define $A = \sum_{i \in T} a_i$. We may assume that $0 < b < A$.
(a) Knapsack $\propto n \mid 2 \mid G, n_i \leq 3 \mid C_{\max}$:

$n = t + 1$;
$\nu_i = (M_1)$, $p_{i1} = a_i$ $(i \in T)$;
$\nu_n = (M_2, M_1, M_2)$, $p_{n1} = b$, $p_{n2} = 1$, $p_{n3} = A - b$;
$y = A + 1$.

If Knapsack has a solution, then there exists a schedule with value $C_{\max} = y$, as illustrated in Fig. 2. If Knapsack has no solution, then $\sum_{i \in S} a_i - b = c \neq 0$ for each $S \subset T$, and we have for a processing order $(\{J_i \mid i \in S\}, J_n, \{J_i \mid i \in T - S\})$ on M_1 that

$$c > 0 \implies C_{\max} \geq \sum_{i \in S} p_{i1} + p_{n2} + p_{n3} = A + c + 1 > y;$$

$$c < 0 \implies C_{\max} \geq p_{n1} + p_{n2} + \sum_{i \in T - S} p_{i1} = A - c + 1 > y.$$

It follows that Knapsack has a solution if and only if this $n \mid 2 \mid G, n_i \leq 3 \mid C_{\max}$ problem has a solution with value $\leq y$.

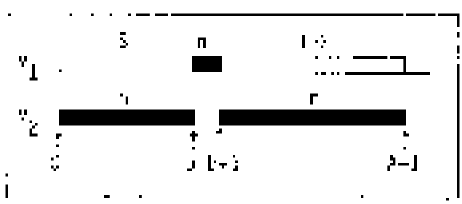


Fig. 2.

(b) Knapsack $\propto n \mid 3 \mid G, n_i \leq 2 \mid C_{\max}$:

$n = t + 2$;
$\nu_i = (M_1, M_3)$, $p_{i1} = p_{i2} = a_i$ $(i \in T)$;
$\nu_{t+1} = (M_1, M_2)$, $p_{t+1,1} = b$, $p_{t+1,2} = 2(A - b)$;
$\nu_n = (M_3, M_1)$, $p_{n1} = 2b$, $p_{n2} = A - b$;
$y = 2A$.

If Knapsack has a solution, then there exists a schedule with value $C_{max} = y$, as illustrated in Fig. 3. If Knapsack has no solution, then $\sum_{i \in S} a_i - b = c \neq 0$ for each $S \subseteq T$, and we have for a processing order $(\{J_i \mid i \in S\}, J_{n+1}, \{J_i \mid i \in T - S\})$ on M_1 that

$$c > 0 \Longrightarrow C_{max} \geq \sum_{i \in S} p_{i1} + p_{n+1,1} + p_{n+1,2} = 2A + c > y,$$

$$c < 0 \Longrightarrow C_{max} \geq \min\Big\{\sum_{i \in S} p_{i1} + p_{n+1,1} - 1, p_{n1}\Big\} + p_{n2} + \sum_{i \in T - S} p_{i2} = 2A - 1 > y,$$

which completes the equivalence proof.

Fig. 3

(e) Knapsack $\propto F3 \| C_{max}$:

$n = t + 1$;
$p_{i1} = 1$, $p_{i2} = ta_i$, $p_{i3} = 1$ $(i \in T)$;
$p_{n1} = tb$, $p_{n2} = 1$, $p_{n3} = t(A - b)$;
$y = t(A + 1) + 1$.

If Knapsack has a solution, then there exists a schedule with value $C_{max} = y$, as illustrated in Fig. 4. If Knapsack has no solution, then $\sum_{i \in S} a_i - b = c \neq 0$ for each $S \subset T$, and we have for a processing order $(\{J_i \mid i \in S\}, J_n, \{J_i \mid i \in T - S\})$ that

$$c > 0 \Longrightarrow C_{max} \geq \sum_{i \in S} p_{i2} + p_{n2} + p_{n3} = t(A + c) + 1 > y;$$

$$c < 0 \Longrightarrow C_{max} \geq p_{n1} + p_{n2} + \sum_{i \in T - S} p_{i2} = t(A - c) + 1 > y.$$

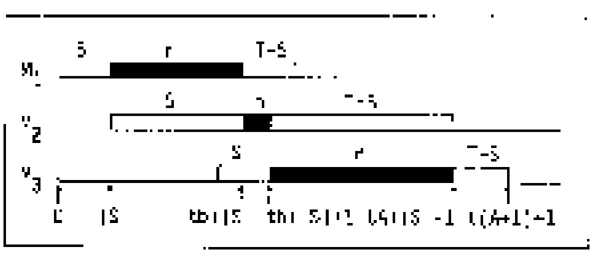


Fig. 4

(d) Knapsack $\propto n|2|F, r_i \geq 0|C_{max}$:

$n = t+1$;
$r_i = 0$, $p_{i1} = ta_i$, $p_{i2} = 1$ $(i \in T)$;
$r_n = tb$, $p_{n1} = 1$, $p_{n2} = t(A-b)$;
$y = t(A+1)$.

Cf. reduction 4(c).

(e) Knapsack $\propto n|2|F|L_{max}$:

$n = t+1$;
$p_{i1} = 1$, $p_{i2} = ta_i$, $d_i = t(A+1)$ $(i \in T)$;
$p_{n1} = tb$, $p_{n2} = 1$, $d_n = t(b+1)$;
$y = 0$.

Cf. reduction 4(e).

(f) Knapsack $\propto n|2|F, tree|C_{max}$:

$n = t+2$;
$p_{i1} = ta_i$, $p_{i2} = 1$ $(i \in T)$;
$p_{t+1,1} = 1$, $p_{t+1,2} = tb$;
$p_{n1} = 1$, $p_{n2} = t(A-b)$;
$J_{t+1} < J_n$;
$y = t(A+1)+1$.

We have for a processing order $(\{J_i \mid i \in R\}, J_{t+1}, \{J_i \mid i \in S\}, J_n, \{J_i \mid i \in T-S-R\})$ on M_1 that

$$R \neq \emptyset \implies C_{max} \geq t + p_{t+1,1} + p_{t+1,2} + p_{n1} + p_{n2} = t(A+1)+2 > y.$$

The remainder of the equivalence proof is analogous to that of reduction 4(e).

(g) Knapsack $\propto n|1|r_i \geq 0|L_{max}$:

$n = t+1$;
$r_i = 0$, $p_i = a_i$, $d_i = A+1$ $(i \in T)$;
$r_n = b$, $p_n = 1$, $d_n = b+1$;
$y = 0$.

Cf. reduction 4(a) and Fig. 5.

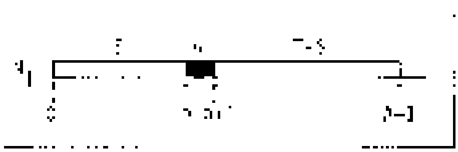


Fig. 5.

(h) Knapsack $\propto n|1|\sum w_i U_i$:

$n = t$;
$p_i = w_i = a_i$, $d_i = b$ $(i \in T)$;
$y = A - b$.

Cf. [25] and Fig. 6.

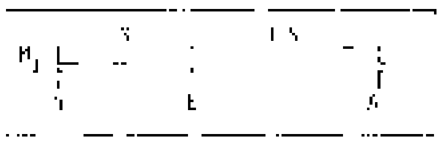


Fig. 6

(i) Knapsack $\propto n\,|\,1\,||\,\sum w_jT_j$:

$$n = t + 1;$$
$$p_j = w_j = a_j,\ d_j = 0\ (j \in T);$$
$$p_n = 1,\ w_n = 2,\ d_n = b + 1;$$
$$y = \sum_{1 \leq i \leq j \leq t} a_i a_j + A - b.$$

Cf. Fig. 5. We have for a processing order $(J_i \,|\, i \in S)$, J_n, $(J_i \,|\, i \in T - S)$ that $\sum_{i\in S} a_i - b = L_n$. Since $p_i = w_i$ and $d_i = 0$ for all $i \in T$, the value of $\sum_{i\in T} w_iT_i$ is not influenced by the ordering of S and $T - S$ (cf. the proof of Theorem 3(b)), and we have

$$\sum w_jT_j = \sum_{i\in T} w_iC_i + 2T_n$$
$$= \sum_{1 \leq i \leq j \leq t} a_i a_j + \sum_{i \in T-S} a_i + 2\max\{0, L_n\}$$
$$= y + |L_n| \geq y.$$

The equivalence follows immediately.

(j) Knapsack $\propto n\,|\,1\,|\,C_j \leq d_j\,|\,\sum w_jC_j$:

$$n = t + 1;$$
$$p_j = w_j = a_j,\ d_j = A + 1\ (j \in T);$$
$$p_n = 1,\ w_n = 0,\ d_n = b + 1;$$
$$y = \sum_{1 \leq i \leq j \leq t} a_i a_j + A - b.$$

Cf. reduction 4(i) and Fig. 5.

This completes the proof of Theorem 4. □

Theorem 5. *3-Partition is reducible to* $n\,|\,1\,|\,r_j \geq 0\,|\,\sum C_j$.

Proof. A reduction 3-Partition $\propto n\,|\,1\,|\,r_j \geq 0\,|\,\sum C_j$ can be obtained by adapting
(a) the transformation of Knapsack to $n\,|\,1\,|\,r_j \geq 0\,|\,\sum C_j$ presented in [45];
(b) the reduction 3-Partition $\propto n\,|\,2\,|\,F\,|\,\sum C_j$ presented in [16].
Both procedures can be carried out in a straightforward way and lead to essentially the same construction. □

The NP-completeness proofs for the problems with a *no wait* assumption are based on the well-known relation between these problems and the travelling salesman problem (TSP) of finding a minimum weight hamiltonian circuit in the complete directed graph on the vertex set V with weights on the arcs.

Given an $n \mid m \mid F, \textit{no wait} \mid \kappa$ problem, we define c_{ij} to be the minimum length of the time interval between S_i and S_j if J_j is scheduled directly after J_i. If we define

$$P_{hk} = \sum_{l=1}^{k} p_{hl} \tag{1}$$

it is easily proved [43; 44; 50; 39] that

$$c_{ij} = \max_{1 \leq k \leq m} \{P_{ik} - P_{j,k-1}\}. \tag{2}$$

Finding a schedule that minimizes C_{max} is now equivalent to solving the TSP with $V = \{0, \ldots, n\}$ and weights c_{ij} defined by (2) and by $c_{0h} = 0$, $c_{h0} = P_{hm}$ for $h \neq 0$.

Theorem 6. Directed hamiltonian path *is reducible to the following problems*:

(a) $n \mid m \mid F, \textit{no wait} \mid C_{max}$;

(b) $n \mid m \mid F, \textit{no wait} \mid \sum C_i$.

Proof.

(a) Directed hamiltonian path $\propto n \mid m \mid F, \textit{no wait} \mid C_{max}$. Given $G' = (V', A')$, we define

$$n = |V'|,$$
$$m = n(n-1) + 2.$$

All jobs have the same machine order $(M_1, M_2, \ldots, M_{m-1}, M_m)$. To each pair of jobs (J_i, J_j) $(i, j = 1, \ldots, n, \; i \neq j)$ there corresponds one machine $M_k = M_{\kappa(i,j)}$ $(k = 2, \ldots, m-1)$, such that for no J_i some $M_{\kappa(\cdot,i)}$ directly follows an $M_{\kappa(i,\cdot)}$. Such an ordering of the pairs (i, j) can easily be constructed. Due to this property of the ordering, partial sums of the processing times can be defined unambiguously by

$$P_{hk} = \begin{cases} k\mu + \lambda & \text{if } k = \kappa(h,j) \text{ and } (h,j) \in A', \\ k\mu + \lambda - 1 & \text{if } k = \kappa(h,j) \text{ and } (h,j) \notin A', \\ k\mu - \lambda & \text{if } k + 1 = \kappa(i,h) \text{ and } (i,h) \in A', \\ k\mu - \lambda - 1 & \text{if } k + 1 = \kappa(i,h) \text{ and } (i,h) \notin A', \\ k\mu & \text{otherwise}, \end{cases}$$

for $k = 1, \ldots, m$, $h = 1, \ldots, n$, where

$$\lambda \geq 1,$$
$$\mu \geq 2\lambda + 3.$$

The processing times are given by (cf. (1))

$$p_{j1} = P_{j1},$$
$$p_{jk} = P_{jk} - P_{j,k-1} \quad (k = 2, \ldots, m).$$

Through the choice of μ, these processing times are all strictly positive integers.

We can now compute the c_{ij} as defined by (2). Through the choice of λ, it is immediate that $P_{ik} - P_{j,k-1}$ is maximal for $k = \kappa(i,j)$. Hence,

$$c_{ij} = \begin{cases} \mu + 2\lambda & \text{if } (i,j) \in A', \\ \mu + 2\lambda - 2 & \text{if } (i,j) \notin A'. \end{cases}$$

Since $P_{jm} = m\mu$ for all J_j, it now follows that G' has a hamiltonian path if and only if this $n \mid m \mid F, \textit{no wait} \mid C_{\max}$ problems has a solution with value

$$C_{\max} \leq (n-1)(\mu + 2\lambda) + m\mu.$$

(b) Directed hamiltonian path $\propto n \mid m \mid F, \textit{no wait} \mid \sum C_j$.
G' has a hamiltonian path if and only if the $n \mid m \mid F, \textit{no wait} \mid \sum C_j$ problem, constructed as in (a), has a solution with value

$$\sum C_j \leq \tfrac{1}{2} n(n-1)(\mu + 2\lambda) + nm\mu. \quad \square$$

Let us finally point out some consequences of the use of a unary encoding with respect to the binary NP-complete problems, appearing in Theorems 3 to 6.

The $n \mid 2 \mid F \mid C_{\max}$ and $n \mid 2 \mid F \mid \sum w_j C_j$ problems, dealt with in Theorem 3, can be solved in unary polynomial-bounded time by straightforward dynamic programming techniques.

A similar situation exists for the $n \mid 1 \| \sum w_j U_j$ problem from Theorem 4(b), which can be solved by an $O(n \sum p_j)$ algorithm [37]. For most other problems discussed in Theorem 4, however, one can easily prove unary NP-completeness by converting the Knapsack reduction to a 3-Partition reduction. The following adaptation of reduction 4(i) might serve as a typical example (cf. the slightly different construction given in [35]).

3-Partition $\propto n \mid 1 \| \sum w_j T_j$:

$$n = 4t - 1;$$
$$p_i = w_i = a_i, \; d_i = 0 \quad (i \in T);$$
$$p_i = 1, \; w_i = 2, \; d_i = (i - 3t)(b + 1) \quad (i = 3t + 1, \ldots, 4t - 1);$$
$$y = \sum_{1 \leq i \leq j \leq 3t} a_i a_j + \tfrac{1}{2} t(t - 1)b.$$

Furthermore, reductions of 3-Partition to $n \mid 2 \mid G \mid C_{\max}$ and $n \mid 3 \mid F \mid C_{\max}$ can be found in [16].

With respect to Theorem 5, the situation is different. In the reductions of 3-Partition to $n \mid 1 \mid r_j \geq 0 \mid \sum C_j$ and $n \mid 2 \mid F \mid \sum C_j$, the resulting numbers of jobs are polynomials in both t and b. The (unary) NP-completeness proofs therefore depend essentially on the unary NP-completeness of 3-Partition and no truly polynomial-bounded transformation of Knapsack to these problems is known.

The reductions presented in Theorem 6 clearly prove unary NP-completeness for both *no wait* problems.

5. Concluding remarks

The results presented in Section 4 offer a reasonable insight into the location of the borderline between "easy" and "hard" machine scheduling problems. Computational experience with many problems proved to be NP-complete confirms the impression that a polynomial-bounded algorithm for one and thus for all of them is highly unlikely to exist. As indicated previously, NP-completeness thus functions as a formal justification to use enumerative methods of solution such as branch-and-bound.

Most classical machine scheduling problems have now been shown to be efficiently solvable or NP-complete. Some notable exceptions are indicated by question-marks in Table 1. These open problem are briefly discussed below.

The most notorious one is the $n|1||\sum T_i$ problem. Extensive investigations have failed to uncover either a polynomial-bounded algorithm or a reduction proving its NP-completeness. The existence of an $O(n^4 \sum p_i)$ algorithm [35] implies that the problem is definitely not *unary* NP-complete. However, we conjecture that it is *binary* NP-complete, which would indicate a major difference between the $\sum T_i$ and $\sum U_i$ problems, as demonstrated by Table 1.

The complexity of the $n|3|F, \textit{no wait}|C_{max}$ and $n|2|F, \textit{no wait}|\sum C_i$ problems is not clear; it is quite possible that both problems are in $\mathscr{P}$. To stimulate research in this direction, we will award an authentic clog to the first scientist who finds a polynomial-bounded algorithm for any one of these problems.

The question of the complexity of the $n|3|I, \textit{prec}, p_{ij} = 1|C_{max}$ problem has been raised already in [49].

Finally, let us stress again that the complexity measure provided by the NP-completeness concept does not capture certain intuitive variations in complexity within the class of NP-complete problems. Note, for example, that an $n|1|r_i \geq 0|L_{max}$ algorithm has figured successfully in a lower bound computation for the $n|m|G|C_{max}$ problem [3;31], although both problems are NP-complete and thus equivalent up to a polynomial-bounded transformation. One possible refinement of the complexity measure by means of differentiation between unary and binary encodings has already been discussed. Another indication of a problem's complexity may be based on the analysis of approximation algorithms [15;27]. For relatively simple NP-complete problems, there often exist heuristics whose performance is arbitrarily close to optimal; on the other hand, there are situations in which even the problem of finding a feasible solution within any fixed percentage from the optimum has been proved NP-complete. Altogether, the development of a measure that allows further distinction within the class of NP-complete problems remains a major research challenge.

Acknowledgements

We gratefully acknowledge the valuable cooperation with B.J. Lageweg, E.L. Lawler and H.W. Lenstra, Jr., and the useful comments by the referee and D.S. Johnson.

References

[1] D. Adolphson and T.C. Hu, Optimal linear ordering, *SIAM J. Appl. Math.* 25 (1973) 403–423.

[2] J.M. Anthonisse and P. van Emde Boas, Are polynomial algorithms really good? Report BW 40, Mathematisch Centrum, Amsterdam, 1974.

[3] P. Bratley, M. Florian and P. Robillard, On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{max})$ problem, *Naval Res. Logist. Quart.* 20 (1973) 57–67.

[4] J. Bruno, E.G. Coffman, Jr. and R. Sethi, Scheduling independent tasks to reduce mean finishing time, *Comm. ACM* 17 (1974) 382–387.

[5] E.G. Coffman, Jr. and R.L. Graham, Optimal scheduling for two-processor systems, *Acta Informat.* 1 (1972) 200–213.

[6] R.W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling* (Addison-Wesley, Reading, MA, 1967).

[7] S.A. Cook, The complexity of theorem proving procedures, *Proc. 3rd Annual ACM Symp. Theory Comput.* (1971) 151–158.

[8] J. Edmonds, Paths, trees, and flowers, *Canad. J. Math.* 17 (1965) 449–467.

[9] J. Edmonds, The Chinese postman's problem, *Operations Res.* 13 Suppl. 1 (1965) B73.

[10] M. Fujii, T. Kasami and K. Ninomiya, Optimal sequencing of two equivalent processors, *SIAM J. Appl. Math.* 17 (1969) 784–789; Erratum, 20 (1971) 141.

[11] M.R. Garey, Private communication, 1975.

[12] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, *SIAM J. Comput.* 4 (1975) 397–411.

[13] M.R. Garey and D.S. Johnson, Scheduling tasks with nonuniform deadlines on two processors, *J. Assoc. Comput. Mach.* 23 (1976) 461–467.

[14] M.R. Garey and D.S. Johnson, Two processor scheduling with start times and deadlines, to appear.

[15] M.R. Garey and D.S. Johnson, Approximation algorithms for combinatorial problems: an annotated bibliography, in: J.F. Traub, ed., *Algorithms and Complexity: New Directions and Recent Results* (Academic Press, New York, 1976) 41–52.

[16] M.R. Garey, D.S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, *Math. Operations Res.* 1 (1976) 117–129.

[17] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified *NP*-complete graph problems, *Theoret. Comput. Sci.* 1 (1976) 237–267.

[18] P.C. Gilmore and R.E. Gomory, Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, *Operations Res.* 12 (1964) 655–679.

[19] W.W. Hardgrave and G.L. Nemhauser, A geometric model and a graphical algorithm for a sequencing problem, *Operations Res.* 11 (1963) 889–900.

[20] W.A. Horn, Single-machine job sequencing with treelike precedence ordering and linear delay penalties, *SIAM J. Appl. Math.* 23 (1972) 189–202.

[21] W.A. Horn, Minimizing average flow time with parallel machines, *Operations Res.* 21 (1973) 846–847.

[22] T.C. Hu, Parallel sequencing and assembly line problems, *Operations Res.* 9 (1961) 841–848.

[23] J.R. Jackson, An extension of Johnson's results on job lot scheduling, *Naval Res. Logist. Quart.* 3 (1956) 201–203.

[24] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Res. Logist. Quart.* 1 (1954) 61–68.

[25] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–103.

[26] R.M. Karp, On the computational complexity of combinatorial problems, *Networks* 5 (1975) 45–68.

[27] R.M. Karp, The fast approximate solution of hard combinatorial problems, *Proc. 6th Southeastern Conf. Combinatorics, Graph Theory, and Computing* (1976) 15–31.

[28] D.E. Knuth, A terminological proposal, *SIGACT News* 6.1 (1974) 12–18.

[29] B.J. Lageweg and E.L. Lawler, Private communication, 1975.

[30] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Minimizing maximum lateness on one machine: computational experience and some applications, *Statistica Neerlandica* 30 (1976) 25–41.

[31] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, Job-shop scheduling by implicit enumeration, *Management Sci.*, to appear.

[32] E.L. Lawler, On scheduling problems with deferral costs, *Management Sci.* 11 (1964) 280–288.

[33] E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints, *Management Sci.* 19 (1973) 544–546.

[34] E.L. Lawler, Sequencing to minimize the weighted number of tardy jobs, *Rev. Française Automat. Informat. Recherche Opérationnelle* 10.5 Suppl. (1976) 27–33.

[35] E.L. Lawler, A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, *Ann. Discrete Math.* 1 (1977) 331–342.

[36] E.L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints, *Ann. Discrete Math.*, to appear.

[37] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, *Management Sci.* 16 (1969) 77–84.

[38] J.K. Lenstra, *Sequencing by Enumerative Methods*, Mathematical Centre Tract 69 (Mathematisch Centrum, Amsterdam, 1977).

[39] J.K. Lenstra and A.H.G. Rinnooy Kan, Some simple applications of the travelling salesman problem, *Operational Res. Quart.* 26 (1975) 717–733.

[40] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operations Res.*, to appear.

[41] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Sci.* 15 (1968) 102–109.

[42] H. Müller-Merbach, *Optimale Reihenfolgen* (Springer, Berlin, 1970).

[43] J. Piehler, Ein Beitrag zum Reihenfolgeproblem, *Unternehmensforschung* 4 (1960) 138–142.

[44] S.S. Reddi and C.V. Ramamoorthy, On the flow-shop sequencing problem with no wait in process, *Operational Res. Quart.* 23 (1972) 323–331.

[45] A.H.G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity and Computations* (Nijhoff, The Hague, 1976).

[46] J.B. Sidney, Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, *Operations Res.* 23 (1975) 283–298.

[47] W.E. Smith, Various optimizers for single-stage production, *Naval Res. Logist. Quart.* 3 (1956) 59–66.

[48] W. Szwarc, Solution of the Akers–Friedman scheduling problem, *Operations Res.* 8 (1960) 782–788.

[49] J.D. Ullman, NP-complete scheduling problems, *J. Comput. System Sci.* 10 (1975) 384–393.

[50] D.A. Wismer, Solution of the flowshop-scheduling problem with no intermediate queues, *Operations Res.* 20 (1972) 689–697.

Annals of Discrete Mathematics 1 (1977) 363-376
© North-Holland Publishing Company

CERTAIN DUALITY PRINCIPLES IN INTEGER PROGRAMMING

L. LOVÁSZ

Bolyai Institute, József Attila University, Szeged, Hungary

This paper surveys some results of the following type: "If a linear program and some derived programs have integral solutions, so does its dual." Several well-known minimax theorems in combinatorics can be derived from such general principles. Similar principles can be proved if integrality is replaced by a condition on the least common denominator of the entries of a solution. An analogy between Tutte's 1-factor-theorem and the Lucchesi–Younger Theorem on disjoint directed cuts is pointed out.

Introduction

The Duality Theorem of linear programming is an extremely useful tool in handling both practical and theoretical problems; here we shall focus on the latter. Whenever a problem can be formulated as a linear program, the Duality Theorem provides us with a re-formulation which often requires a mere computation to solve; and in all cases it gives a new insight into the problem. Those problems arising from combinatorics have in most cases, the additional constraint that the variables are restricted to integers. Many — and often the deepest — results in combinatorics assert that for certain classes of integer programs the Duality Theorem remains valid. These facts tempt one to try to develop general methods in integer programming which would enable us to handle different minimax results in combinatorics together. The Hoffman–Kruskal Theorem on unimodular matrices, Edmonds' theory of matchings, Berge's theory of balanced hypergraphs and Fulkerson's theory of blocking and anti-blocking polyhedra represent results in this direction.

The aim of this paper is to survey some results and applications in the above-mentioned direction. Most of the general theorems will be of the following form: if an integer program and certain derived programs have nice solutions (e.g. the same solutions as if they were considered as linear programs) then so do their duals. Very often it is useful to consider not only integral and real solutions but also those where the denominators of coordinates are restricted to divisors of an integer k. For $k = 2$, this links a variety of solved and unsolved graph-theoretical problems to our duality results.

The whole area is not yet worked out very well. There are several minimax results in graph theory which almost fit into this pattern but their generalization to integer programming has not yet been found (at least not in the spirit of our paper). Also, there are several open problems, which will be formulated in the paper. Most of the theorems are formulated very similarly but their proofs are based on completely different ideas (this was also remarked by A.J. Hoffman at the International Congress of Mathematicians in Vancouver). This probably shows that our understanding of the matter is superficial.

Only a few proofs will be given in detail: those which have not yet been published and seem to be characteristic.

Definitions and notations

We will restrict ourselves to packing and covering programs; combinatorial problems transform almost always into such programs. We shall use the language of hypergraphs rather than of matrices. This is more difficult to compute with but makes things easier to visualize.

A hypergraph H is a finite collection of non-empty finite sets; the same set may occur more than once. The elements of hypergraphs are called edges, the elements of edges are *points* (this way no isolated points are allowed). The set of vertices of the hypergraph H will be denoted by $V(H)$.

Removing an edge means that we remove this edge and all points which would become isolated by this. *Removing a vertex* $x \in V(H)$ means that we remove all edges adjacent to x. *Multiplying* a vertex x by $k \geq 0$ means that we replace x by k points $x_1, \ldots, x_k$ and replace each edge E containing x by k edges $E - \{x\} \cup \{x_i\}$, $i = 1, \ldots, k$.

Given a hypergraph H, we are interested in the maximum number, $\nu(H)$, of disjoint edges of H and in the minimum number, $\tau(H)$, of points representing all edges of H. To study these numbers we will introduce some related numbers.

A *k-matching* is a mapping $m : H \to \{0, 1, \ldots\}$ such that

$$\sum_{E \ni x} m(E) \leq k \quad (x \in V(H)).$$

A *k-cover* is a mapping $t : V(H) \to \{0, 1, \ldots\}$ such that

$$\sum_{x \in E} t(x) \geq k \quad (E \in H).$$

A *fractional matching* is a mapping $m : H \to \{\text{non-negative reals}\}$ such that

$$\sum_{E \ni x} m(E) \leq 1 \quad (x \in V(H)),$$

and a *fractional cover* is a mapping $t : V(H) \to \{\text{non-negative reals}\}$ such that

$$\sum_{x \in E} t(x) \geq 1 \quad (E \in H).$$

We shall denote by $\nu_k(H)$ and $\tau_k(H)$ the maximum of

$$\|w\| = \sum_{E \in H} w(E)$$

for all k-matchings and the minimum of

$$\|t\| = \sum_{x \in V(H)} t(x)$$

for all k-covers, respectively. If w runs over all fractional matchings and t runs over all fractional covers we have

$$\max \sum_{E \in H} w(E) = \min \sum_{x \in V(H)} t(x) \stackrel{\text{def}}{=} \tau^*(H),$$

by the Duality Theorem.

It is trivial that $\tau_1 = \tau$, $\nu_1 = \nu$ and also that

$$\nu \leq \frac{\nu_k}{k} \leq \tau^* \leq \frac{\tau_k}{k} \leq \tau$$

for every k. There is an integer s such that

$$\nu_{ks} = ks\tau^*, \quad \tau_{ks} = ks\tau^*$$

for every k (since the linear programs defining τ^* have rational optimal solutions).

Duality results

The following two theorems were proved in [8]; they can be derived from the theory of blocking and anti-blocking polyhedra as well [3]:

Theorem 1. *If $\nu(H') = \tau^*(H')$ holds for every hypergraph H' obtained from H by removing points then $\nu(H) = \tau(H)$.*

Theorem 2. *If $\tau(H') = \tau^*(H')$ holds for every $H' \subseteq H$, then $\nu(H) = \tau(H)$.*

Berge [1] has observed the following sharpening of Theorem 2:

Theorem 2'. *If $\tau_2(H') = 2\tau(H')$ holds true for every $H' \subseteq H$, then $\nu(H) = \tau(H)$.*

In [9] it was shown that if we require the inheritence for a larger class of hypergraphs then the assumption in Theorem 2 can be weakened analogously:

Theorem 1'. *If $\nu_2(H') = 2\nu(H')$ holds for every hypergraph H' arising from H by multiplication of vertices, then $\nu(H) = \tau(H)$.*

We remark that in the case of graphs the following much simpler result holds:

Theorem 3. *Let G be a graph. Then $\tau_2(G) = 2\tau(G)$ implies $\nu(G) = \tau(G)$. More generally, for any graph we have $\tau_2(G) \le \tau(G) + \nu(G)$.*

Let us remark that in the last three assertions replacing the index 2 by an arbitrary index k we could obtain similar results, which would be trivial consequences of those formulated above.

Let us formulate two assertions:

(*) If $\nu_k(H') = k\tau^*(H')$ holds for each hypergraph H' arising from H by multiplication of vertices then $\nu_k(H) = \tau_k(H)$.

(**) If $\nu_k(H') = k\tau^*(H')$ holds true for each $H' \subset H$, then $\nu_k(H) = \tau_k(H)$.

These are true for $k = 1$ by Theorems 1' and 2. In [10] they were proved for $k = 2$. Since my proof completely failed to work for $k = 3$, I ventured to conjecture that they were false. However, recently I have found a proof of (**) for $k = 3$. As my proof is rather complicated and it does not generalize to $k = 4$, I only dare to set it as a question: Are (*) and (**) valid for other, maybe for all, values of k?

Theorem 4. *Let $k = 1, 2$ or 3. Let H be a hypergraph such that $k \cdot \tau^*(H')$ is an integer for all $H' \subseteq H$. Then $\nu_k(H) = k \cdot \tau^*(H)$.*

This clearly implies (**). We remark that the analogous sharpening of (*) is also valid for $k = 1, 2$.

Proof of Theorem 4. Let H be a minimal counterexample and m an optimal fractional matching of H. Since

$$\nu_k(H) < k \cdot \tau^*(H)$$

but

$$\nu_k(H - \{E\}) = k \cdot \tau^*(H - \{E\})$$

for any $E \in H$, it follows that

$$k\tau^*(H) \ge \nu_k(H) + 1 \ge \nu_k(H - \{E\}) + 1 = k\tau^*(H - \{E\}) + 1$$

or

$$\tau^*(H - \{E\}) \le \tau^*(H) - \frac{1}{k}.$$

Since $m' = m|_{H-\{E\}}$ is a fractional matching of $H - \{E\}$, we must have

$$|m'| = |m| - m(E) \le \tau^*(H - \{E\}) \le \tau^*(H) - \frac{1}{k},$$

whence

$$m(E) \ge \frac{1}{k}.$$

For $k = 1$ this implies $m(E) = 1$ hence that m is a 1-matching, $\nu_1(H) = \tau^*(H)$. So in this case the proof is finished.

Observe now that if x is any point with degree $\geq k$ then

$$\sum_{E \ni x} m(E) \leq 1$$

can only be fulfilled if the degree of x is k and $m(E) = 1/k$ for all edges incident to x. Also we may suppose each edge E contains a point of degree ≥ 2 as otherwise its removal would decrease both ν_k and $k\tau^*$ by exactly k, and $H - \{E\}$ would be a smaller counterexample. Hence if $k = 2$ then $m(E) = 1/2$ for all edges and we are finished again.

So suppose $k = 3$. We show H has a point of degree 3. Suppose not. The intersection graph $L(H)$ of H cannot be bipartite, since then H would be balanced (see [1]) and $\nu_k(H) = k \cdot \tau^*(H)$ would follow from the much stronger relation $\nu(H) = \tau(H)$. Let $(E_1, \ldots, E_{2p+1})$ be any chordless odd circuit in $L(H)$. Then $H' = \{E_1, \ldots, E_{2p+1}\}$ has $3\tau^*(H') = 3p + \frac{3}{2}$, contradicting the assumption that this should be an integer.

Thus H has a point x_0 of degree 3. Let E_0 be any edge adjacent to x_0. Then we know $m(E_0) = 1/3$. We need the following

Lemma. *Let H be any hypergraph and $E_0 \in H$. Then H has a decomposition $H = H_1 \cup H_2$ and H_1 has an optimum fractional matching m_1 with the following properties:*

(i) $E_0 \subset H_1$;

(ii) *for any decomposition $H_1 = H_1' \cup H_1''$, $H_1', H_1'' \neq \emptyset$ there is an $x \in V(H_1') \cap V(H_1'')$ with*

$$\sum_{E \ni x} m_1(E) = 1;$$

(iii) *for any optimum fractional matching m_2 of H_2, $m_1 \cup m_2$ is an (optimum) fractional matching of H.*

Supposing this Lemma is true, let E_0 be an edge adjacent to a point of degree 3 and consider the decomposition $H_1 \cup H_2 = H$ defined in the Lemma. The fractional matching m_1 of H_1 takes values 1/3, 2/3 only. For let

$$H_1' = \{E : m_1(E) = 1/3 \text{ or } 2/3\}$$

$$H_1'' = H_1 - H_1'.$$

Then $E_0 \in H_1'$ and so $H_1' \neq \emptyset$. If $H_1'' \neq \emptyset$ then by (ii), there is a point $x \in V(H_1') \cap V(H_1'')$ with

$$\sum_{E \ni x} m_1(E) = 1.$$

Let $E_1 \subset H_1'$, $E_2 \subset H_1''$, $x \subset E_1 \cap E_2$. If x has degree 3 then, as noted above, $m_1(E_2) = 1/3$. If x has degree 2 then

$$m_1(E_2) = 1 \quad m_1(E_2) = 1/3 \text{ or } 2/3$$

since $E_2 \subseteq H_1'$. In both cases we get a contradiction with $E_2 \subseteq H_1'$.

So m_1 takes values 1/3 and 2/3. By the minimality assumption on H, H_2 has an optimum fractional matching m_2 whose values are 0, 1/2, 2/3 or 1. By (iii) of the Lemma, $m_1 \cup m_2$ is an optimum fractional matching with values 0, 1/2, 2/3 or 1. This proves $\nu_3(H) = 3\nu^*(H)$, a contradiction.

Proof of the Lemma. Choose an $H_1 \subseteq H$ and a maximum fractional matching m_0 of H such that with $m_1 = m_0 | H_1$, (i) and (ii) are fulfilled. E.g. $H_1 = \{E_0\}$ is such a partial hypergraph; but choose H_1 maximal among all subcollections of H for which an m_1 exists satisfying (i) and (ii). Let $H_2 = H - H_1$.

Then for every edge $E \in H_2$, and for each point $x \in E \cap V(H_1)$,

$$\sum_{F \ni x} m_0(F) < 1; \tag{1}$$

otherwise E could be added to H_1. Let m_2 be any optimum fractional matching of H_2, we claim $m_1 \cup m_2$ is a fractional matching. For let ε be a sufficiently small positive number and

$$m'(F) = \begin{cases} m_0(F) & \text{for } F \in H_1, \\ \varepsilon m_2(F) + (1-\varepsilon) m_0(F) & \text{for } F \in H_2. \end{cases}$$

Then by (1), $m'(F)$ is a fractional matching if ε is small enough. Hence

$$|m'| \leq |m_0|$$

or

$$\sum_{E \in H_2} m_2(E) \leq \sum_{E \in H_2} m_0(E).$$

But since m_2 is an optimum fractional matching of H_2, we also have here the converse inequality. Hence $|m'| = |m_0|$, i.e. m' is an optimum fractional matching.

If m' is a fractional matching for $\varepsilon = 1$, then $m_1 \cup m_2$ is a fractional matching as claimed. So suppose there is a largest ε_0, $0 < \varepsilon_0 < 1$ for which m' is a fractional matching. Then there must be a point $x \in V(H_1) \cap V(H_2)$ such that

$$\sum_{F \ni x} m'(F) = 1.$$

Thus replacing m_0 by m', H_1 can be enlarged. This contradiction proves the assertion that $m_1 \cup m_2$ is a fractional matching. Now the optimality of $m_1 \cup m_2$ is clear since

$$|m_1 \cup m_2| = |m_1| + |m_2| \geq |m_0 | H_1| + |m_0 | H_2| = |m_0|.$$

This finishes the proof of the Lemma.

Decompositions

To guarantee that the conditions of the preceding theorems hold one has to show that for hypergraphs arising from given combinatorial structures, fractional covers with denominator k are not any better than fractional covers with denominator j, for certain values of k and j. (The derived hypergraphs arise usually in the same way, so they need not be considered extra.) This often depends on the fact that multiple covers decompose into the sum of other multiple covers.

Suppose a k-cover (k-matching) is the sum of a k_1-cover and a k_2-cover (matching). If $k_1 + k_2 = k$, we call this decomposition *exact*. The following two theorems can be proved by a straightforward construction.

Theorem 5. *Let G be a bipartite graph. Then each k-cover of G can be exactly decomposed into a sum of 1-covers.*

Theorem 6. *Let G be a graph. Then each k-cover of G can be (exactly) decomposed into the sum of a 2-cover and a $(k-2)$-cover.*

A construction due to R.L. Graham [5] shows that no analogue of Theorem 6 is valid for 3-uniform hypergraphs: there may be exactly indecomposable k-covers for arbitrary large k.

An *r-partite hypergraph* ($r \geq 2$) is defined as follows: $V(H)$ has a partition $V_1 \cup \cdots \cup V_r$ such that each edge meets each V_i in exactly 1 point. Not even k-covers of r-partite hypergraphs are always exactly decomposable for $r \geq 3$; but for non-exact decompositions, we have the following results:

Theorem 7. *Each k-cover of an r-partite hypergraph is a sum of $1 + \lfloor 2(k-1)/r \rfloor$ 1-covers.*

Theorem 8. *Each k-cover of an r-uniform hypergraph is sum of an r-cover and a $(k - 2(r-1))$-cover. Also, each k-cover is the sum of a 2-cover and a $(k-r)$-cover.*

Proof. We only give the proof of Theorem 7; Theorem 8 can be verified by similar direct constructions.

We need a

Lemma. *Let $r \geq 2$, $m \geq 0$. Then there exists an $r \times (m+1)$ matrix (a_{ij}) such that*

(a) *each row is a permutation of $(0, 1, \ldots, m)$;*

(b) $\sum_{i=1}^{r} a_{ij} \leq \lceil \frac{1}{2} r \cdot m \rceil^*$ *for $j = 1, 2, \ldots, m+1$ ($\lceil x \rceil^*$ is the least integer $\geq x$).*

Proof. If we have such a matrix then we can get one for $r+2$ by adding two rows

$$(0 \quad 1 \quad \cdots \quad m)$$

$$(m \quad m-1 \quad \cdots \quad 0).$$

So it suffices to deal with the cases $r = 2, 3$. For $r = 2$,

$$\begin{bmatrix} 0 & 1 & \cdots & m \\ m & m-1 & \cdots & 0 \end{bmatrix}$$

is an appropriate matrix. If $r = 3$ and m is even then

$$\begin{bmatrix} 0 & 1 & \cdots & \frac{m}{2} & \frac{m}{2}+1 & \cdots & m \\ \frac{m}{2} & \frac{m}{2}+1 & \cdots & m & 0 & \cdots & \frac{m}{2}-1 \\ m & m-2 & \cdots & 0 & m-1 & \cdots & 1 \end{bmatrix}$$

if $r = 3$ and m is odd then

$$\begin{bmatrix} 0 & 1 & \cdots & \frac{m-1}{2} & \frac{m+1}{2} & \cdots & m-1 & m \\ \frac{m+1}{2} & \frac{m+3}{2} & \cdots & m & 0 & \cdots & \frac{m-3}{2} & \frac{m-1}{2} \\ m & m-2 & \cdots & 1 & m-1 & \cdots & 2 & 0 \end{bmatrix}$$

is an appropriate matrix.

Let, now, t be a k-cover of an r-partite hypergraph H and define

$$t_i(x) = \begin{cases} 1 & \text{if } x \in V_j \text{ and } t(x) \geq a_{ji}+1, \\ 0 & \text{otherwise,} \end{cases}$$

where $m = \lfloor 2(k-1)/r \rfloor$, a_{ij} is defined as in the Lemma and $i = 0, \ldots, m$. Then for $x \in V_j$

$$\sum_{i=0}^{m} t_i(x) = \sum_{a_{ji} < t(x)} 1 = \min(t(x), m)$$

and thus,

$$\sum_{i=0}^{m} t_i \leq t.$$

On the other hand, we claim each t_i defines a 1-cover. For let $(v_1, \ldots, v_r) \in H$, $v_j \in V_j$. Suppose for some i,

$$t_i(v_j) = 0 \quad \text{for } j = 1, \ldots, r.$$

This means

$$t(v_j) \leq a_{ji}$$

and hence

$$\sum_{i=1}^{r} t(x_i) \le \sum_{i=1}^{r} a_i = \left[\frac{mr}{2}\right]^{*} < k,$$

a contradiction. This proves the Theorem.

The following are examples of decomposition results for certain special hypergraphs of interest.

Theorem 9. *Let G be a graph and a, b two specified points. Consider the sets of edges of (a, b)-paths in G as edges of a hypergraph H. Then any k-cover of H can be exactly decomposed into 1-covers.*

Theorem 10. *Let G be a graph, $S \subseteq V(G)$. Call a path P principal if its endpoints are in S but has no inner point in S. The sets of edges of principal paths form a hypergraph H. Then each k-cover of H can be (exactly) decomposed into a 2-cover and a $(k-2)$-cover.*

Theorem 11. *Let G be a digraph, a a specified vertex (root) and consider the sets of edges of spanning arborescences rooted at a as edges of a hypergraph H. Then each k-cover of H can be exactly decomposed into 1-covers.*

In all three cases, there is a direct construction proving them (direct not meaning simple). Theorems 9 and 11 are also consequences of results of Fulkerson [3, 4].

Proof. As an example we give that of Theorem 10. Let t be a k-cover of H; define

$$t_1(e) = \begin{cases} 2 & \text{if } t(e) \ge k; \\ 1 & \text{if } 0 < t(e) < k \text{ and } e \text{ is the first edge with } t(e) > 0 \text{ on one} \\ & \text{of the principal paths (starting from either endpoint);} \\ 0 & \text{otherwise;} \end{cases}$$

$t_2(e) = t(e) - t_1(e)$.

It is immediate to see that t_1 is a 2-cover of H. To show that t_2 is a $(k-2)$-cover consider principal paths P such that

$$\sum_{e \in P} t_1(e)$$

is minimal and among these

$$\sum_{e \in P} t(e)$$

is minimal. We claim P contains at most two edges e with $t_1(e) = 1$. For if there were three such edges, e_1, e_2, e_3 say, in this order on the path P, then there would be a path Q connecting a point of S to an endpoint of e_2 such that $t(f) = 0$ for $f \in Q$. This path Q, together with one half of P, would form a principal path P' such that

$$\sum_{e\in P'} t_2(e) \le \sum_{e\in P} t_2(e),$$

$$\sum_{e\in P'} t(e) < \sum_{e\in P} t(e),$$

a contradiction.

Now if $t_2(f) = 2$ for some $f \in P$, then

$$\sum_{e\in P} t_2(e) \ge t_2(f) \ge k-2,$$

otherwise

$$\sum_{e\in P} t_2(e) = \sum_{e\in P} t(e) - \sum_{e\in P} t_1(e) \ge k-2.$$

This proves the Theorem.

The relation $\tau_k = k\tau$ is clearly a consequence of, but not equivalent to, the fact that k-covers are exactly decomposible into 1-covers. An example showing that $\tau_k = k\tau$ does not imply the decomposability of 1-covers is yielded by the following.

Theorem 12. *Let C be a chain group* mod 2 *on a set S of atoms and let C' be a coset of C. Considering the non-zero elements of C' as subsets of S, the resulting hypergraph H satisfies $\tau_2(H) = 2\tau(H)$.*

As an example of a hypergraph obtainable as in this theorem, consider the hypergraph whose edges are the sets of edges of odd circuits in a graph. Here $t = 1$ represents a 2-cover (even a 3-cover) which is not the sum of two 1-covers if the chromatic number of the graph is larger than 4 [14].

The exact decompositions of matchings are, usually, more difficult to handle. The following results are true, but their proofs are not "direct": they follow from well-known theorems of König and Petersen, respectively.

Theorem 13. *Let G be a bipartite graph. Then each k-matching of G is the sum of k 1-matchings.*

Theorem 14. *Let G be a graph. Then each $(2k)$-matching of G is the sum of k 2-matchings.*

Often one can replace a k-matching by another k-matching of the same size and of simpler structure, which can be decomposed. E.g. if the edges are cuts of a graph or digraph, one can replace crossing cuts by non-crossing (laminar) ones (see Lucchesi and Younger [17]). The following two theorems can be proved this way:

Theorem 15. *Let H consist of the directed cuts of a digraph. Then $\nu_2(H) = 2\nu(H)$.*

Theorem 16. *Let G be a graph and $A \subseteq V(G)$, $|A|$ even. Let H consist of those cuts of G having an odd number of points of A on both sides. Then $\nu_4(H) = 2\nu_2(H)$.*

Similar manipulation with paths yields

Theorem 17. *Let us mark each point of a graph G by 1, 1', 2 or 2'. Let H consist of all paths connecting a 1 to a 1' or a 2 to a 2'. Then $\nu_2(H) = 2\tau(H)$.*

Examples

Putting results of the two previous sections together we obtain several minimax results in graph theory. Thus Theorems 2 and 5 yield König's Theorem (this theorem could be deduced from any of theorems 1', 2, 3 easily). Similarly, Theorems 4 (with $k = 2$) and 6 imply the fact, observed by J. Edmonds and probably others that each graph satisfies $\nu_2 = \tau_2$. Theorems 1' and 15 imply the result of Lucchesi and Younger (conjectured for several years by Robertson and Younger) that the minimum number of edges in a digraph whose contraction results in a strongly connected digraph equals the maximum number of edge-disjoint directed cuts. Theorems 1' and 17 imply a theorem of Kleitman, Martin-Löf, Rothschild and Whinston [6].

Putting Theorems 16 and (*) for $k = 2$ together we obtain a result which is probably new [10]; and whose proof is given as a typical example of arguments here.

Theorem 18. *Let k denote the minimum number of edges of a graph G on $2n$ points such that the subgraph on $V(G)$ formed by them has even components. Then the maximum number of cuts, separating G into two odd pieces and containing each edge at most twice, is $2k$.*

Considering the case when $k = n$, Tutte's Theorem on 1-factors can be deduced.

Proof. Let H be the hypergraph whose edges are those cuts of G which have an odd number of points on both sides. It is easy to see that each hypergraph arising from H by multiplying vertices is of the type considered in Theorem 16, and so, it satisfies $\nu_4(H') = 2\nu_2(H')$. Thus the conditions of (*) (for $k = 2$) are fulfilled and hence, $\nu_2(H) = \tau_2(H)$. Moreover, H arises as the hypergraph in Theorem 12 (as a coset of the chaingroup of all cuts separating the graph into two even pieces), and so, Theorem 12 implies $\tau_2(H) = 2\tau(H)$. Thus $\nu_2(H) = 2\tau(H)$. Now a set of edges of G covers all edges of H iff the subgraph formed by them on $V(G)$ has even components. So $\tau(H) = k$ and $\nu_2(H) = 2k$, which is the assertion of the theorem.

Finally we quote three theorems which can be compared with Theorems 9–11 though no general result on hypergraphs would reduce them to those.

Theorem 19 (Menger's Theorem). *The hypergraph in Theorem* 9 *satisfies* $\nu = \tau$.

Theorem 20 (J. Edmonds [2]). *The hypergraph in Theorem* 10 *satisfies* $\nu = \tau$.

Theorem 21 [11]. *The hypergraph in Theorem* 11 *satisfies* $\nu_2 = \tau_2$.

Problems. It is immediate to ask for duality principles linking Theorems i and $10+i$ for $i = 9, 10, 11$.

It may be a very widely applicable direction of generalization of these problems to extend them to situations where exact equalities are replaced by inequalities (going in the non-trivial direction, of course).

There are very many results and problems in combinatorics asserting that for certain hypergraphs ν and τ^* or ν^* and τ are "close" to each other. Just to mention a few: The Edmonds–Tutte theorem on disjoint bases of a matroid; Vizing's Theorem; Gallai's conjecture that in the hypergraph in Theorem 10, $\tau \leq 2\nu$, etc. Finding results for such "loose" situations would be very useful.

A conjecture of Ryser could be formulated as follows: each r-partite hypergraph satisfies $\tau \leq (r-1)\nu$. Our Theorem 7 implies $\tau^* \leq \frac{1}{2}r\nu$. Is there any duality principle which would allow us to deduce Ryser's conjecture from this?

References

[1] C. Berge, Balanced hypergraphs and some applications to graph theory, in: J.N. Srivastava, ed., A *Survey of Combinatorial Theory* (North-Holland, Amsterdam, 1973) 15–23.

[2] J. Edmonds, Submodular functions, matroids, and certain polyhedra, in: Comb. Structures Appl. (Gordon and Breach, London, 1970) pp. 69–87.

[3] D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra, Math. *Programming* 1 (1971) 168–194.

[4] D.R. Fulkerson, Packing weighted directed cuts in rooted directed graphs, Math. *Programming* 6 (1974) 1–13.

[5] R.L. Graham, 3-uniform hypergraphs can have horrible covers (preprint).

[6] D. Kleitman, A. Martin-Löf, B. Rothschild and A. Whinston, A matching theorem for graphs, J. Comb. Theory 8 (1970) 104–114.

[7] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.* 2 (1972) 253–267.

[8] L. Lovász, Minimax theorems for hypergraphs, *Hypergraph Seminar*, Lecture Notes in Math. 411 (1974) 111–126.

[9] L. Lovász, On two minimax theorems in graph theory, *J. Comb. Theory* B 21 (1976) 96–103.

[10] L. Lovász, 2-matchings and 2-covers of hypergraphs, *Acta Math. Acad. Sci. Hung.* 26 (1975) 433–444.

[11] L. Lovász, On some connectivity properties of Eulerian graphs, *Acta Math. Acad. Sci. Hung.* 28 (1976) 129–138.

[12] C. Lucchesi and D.H. Younger, A minimax theorem for directed graphs, *Proc. London Math. Soc.* (to appear).

[13] D.H. Younger, Maximum families of disjoint directed cut sets, *Recent Progress in Combin.*, Academic Press, 1969, 329–333.

[14] D.R. Woodall, Property B and the four color conjecture, *Combinatorics*, Proc. Conf. Southend-on-Sea (1972) 322–340.

Annals of Discrete Mathematics 1 (1977) 375-390

PARAMETRIC INTEGER PROGRAMMING: THE RIGHT-HAND-SIDE CASE

Roy E. MARSTEN

Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Thomas L. MORIN

School of Industrial Engineering, Purdue University, West Lafayette, IN, U.S.A.

A family of integer programs is considered whose right-hand-sides lie on a given line segment L. This family is called a parametric integer program (PIP). Solving a (PIP) means finding an optimal solution for every program in the family. It is shown how a simple generalization of the conventional branch-and-bound approach to integer programming makes it possible to solve such a (PIP). The usual bounding test is extended from a comparison of two point values to a comparison of two functions defined on the line segment L. The method is illustrated on a small example and computational results for some larger problems are reported.

Acknowledgement. The computer implementation of the algorithm reported here was performed by Lee Aurich and Nancy Kaplan.

1. Introduction

The purpose of this paper is to show how a simple generalization of the conventional branch and bound approach to integer programming makes it possible to solve a parametric integer program. Following Nauss [6] we shall call the family of programs (P_θ)

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \leq b_i + \theta d_i \quad 1 \leq i \leq m$$

$$x_j \in \{0, 1\} \quad 1 \leq j \leq n$$

for $0 \leq \theta \leq 1$ a single parametric integer program (PIP). By "solving" (PIP) we shall mean obtaining an optimal solution of (P_θ) for every $0 \leq \theta \leq 1$ for which (P_θ) is feasible. We assume that (P_θ) is feasible for at least one value of θ.

Parametric integer programming has only recently emerged as a topic of research. The pioneering papers include Noltemeier [7], Roodman [10, 11], Piper and Zoltners [8, 9], and Bowman [1]. Nauss [6] has reviewed this earlier work and contributed many new results for parameterizations of the objective function. The present paper, which has grown out of the authors' work on synthesizing dynamic programming with branch-and-bound [3, 4, 5], is devoted to the right-hand-side case.

In parametric *linear* programming, the first step is to solve (P_0), i.e. (P_θ) for $\theta = 0$. Then the direction vector $d = (d_1, \ldots, d_m)$ is specified and the analysis is performed by driving θ from 0 to 1. Critical values of θ and new optimal solutions are identified one at a time as θ increases. In the procedure for parametric *integer* programming to be presented here, the direction d must be specified in advance. The (PIP) is solved in one branch-and-bound search. The usual bounding test is modified so that a partial solution is eliminated only if none of its descendants is optimal for any (P_θ), $0 \leq \theta \leq 1$. This means that some partial solutions must be retained that could otherwise be eliminated if only (P_0) were of interest. The severity of the resulting computational burden depends on the magnitude of d.

The organization of the paper is as follows. A prototype branch-and-bound algorithm for (P_0) is presented in Section 2.
The lower bound and upper bound functions are developed in Sections 3 and 4, respectively. The modified branch-and-bound algorithm for (PIP) is given in Section 5 and applied to a sample problem in Section 6. Computational experience with the algorithm is reported in Section 7.

2. A prototype branch-and-bound algorithm

We shall draw upon the framework and terminology of Geoffrion and Marsten [2] to describe a simple linear programming based branch-and-bound algorithm for (P_0). Problem (P_0) is separated, by fixing variables at zero and one, into smaller candidate problems (CP^q). Each candidate problem has an associated set of fixed variables $F^q \subseteq J = \{1, \ldots, n\}$ and partial solution x^q. That is, (CP^q) is defined by the conditions $x_j = x_j^q$ for $j \in F^q$. The current set of candidate problems is called the candidate list. If any feasible solutions of (P_0) are known, the best of these is called the incumbent and its value denoted by LB. If we let $J^q = J - F^q$ be the set of "free" variables and

$$\beta^q = \sum_{j \in F^q} A_j x_j^q$$

where A_j is the jth column of A, then a typical candidate problem may be written as (CP^q)

$$\sum_{j \in F^q} c_j x_j^q + \max \sum_{j \in J^q} c_j x_j$$

$$\text{subject to} \sum_{j \in J^q} a_{ij} x_j \leq b_i - \beta_i^q, \quad 1 \leq i \leq m,$$

$$x_j \in \{0, 1\}, \quad j \in J^q$$

An upper bound on the value of (CP^q) is obtained by solving its LP relaxation (CP^q_R). It is also helpful to compute a lower bound on the value of (CP^q). This can be done by using a heuristic to find a feasible solution of (CP^q). This feasible solution, if it is better than the incumbent, becomes the new incumbent. A prototype branch-and-bound algorithm may now be described as follows.

Step 1. Place (P_0) in the candidate list and set $LB = -\infty$.

Step 2. If the candidate list is empty, stop. If there is an incumbent, it is optimal for (P_0). Otherwise (P_0) is infeasible.

Step 3. Select a candidate problem and remove it from the candidate list. Call it (CP^q).

Step 4. Solve the linear program (CP^q_R). Let UB^q denote its optimal value.

Step 5. If $UB^q \leq LB$, go to Step 2.

Step 6. If the optimal solution of (CP^q_R) is all integer, make this solution the new incumbent, set $LB = UB^q$, and go to Step 2.

Step 7. Use a heuristic to find a feasible solution of (CP^q). Let H^q denote its value. If $H^q > LB$, then make this solution the new incumbent and set $LB = H^q$.

Step 8. Separate (CP^q) into two candidate problems $(CP^{q'})$ and $(CP^{q''})$ by choosing $p \in F^q$ and setting $F^{q'} = F^{q''} = F^q - \{p\}$, $x^{q'}_p = 0$, $x^{q''}_p = 1$. Place $(CP^{q'})$ and $(CP^{q''})$ in the candidate list and return to Step 2.

A great many variations on this pattern are described in [2], but this prototype will suffice for our purposes. Step 5 is the bounding test. If this test is satisfied, then no descendant of x^q is better than the incumbent. Notice that the bounding test includes the case where (CP^q_R), and hence (CP^q), is infeasible since then $UB^q = -\infty$. If (CP^q) does not have to be separated at Step 8, then we say that it has been fathomed. This occurs if (CP^q) passes the bounding test or if (CP^q_R) has an all integer solution. Step 7, the heuristic, is optional. Its purpose is to strengthen the bounding test by improving the incumbent and increasing LB.

The modifications that must be made to this prototype algorithm to solve (PIP) are confined to Steps 5, 6 and 7. The notion of the incumbent must be generalized from a single value LB to a function $LB(\theta)$ defined on $0 \leq \theta \leq 1$. The upper bound must also be expressed as a function of θ: $UB^q(\theta)$. The bounding test then becomes a comparison of two functions on the interval $0 \leq \theta \leq 1$ rather than just a point comparison for $\theta = 0$.

3. The optimal return and lower bound functions

In this section we shall investigate the behavior of the optimal value of an integer program as a function of its right-hand-side. Let the optimal return function

$$
\begin{aligned}
f(b') = \max\ & cx \\
\text{subject to } & Ax \leq b' \\
& x \in \{0, 1\}
\end{aligned}
$$

be defined for $b' \in \mathbf{R}^m$. It is apparent that $f(b')$ is nondecreasing in each component of b'. Let $\{x^k \mid k \in K\}$ be the set of all feasible solutions of (PIP), i.e. of all (P_θ) for $0 \le \theta \le 1$. For each $k \in K$, define the step function

$$f^k(b') = \begin{cases} \sum_{j=1}^{n} r_j x_j^k & \text{if } b' \ge \sum_{j=1}^{n} A_j x_j^k \\ -\infty & \text{otherwise} \end{cases}$$

for all $b' \in \mathbf{R}^m$. The optimal return function $f(b')$ is the pointwise maximum of this finite collection of nondecreasing step functions

$$f(b') = \max\{f^k(b') \mid k \in K\}$$

and is therefore itself a nondecreasing step function.

Now suppose that the solutions $\{x^k \mid k \in \bar{K}\}$ are known, where $\bar{K} \subseteq K$. A lower approximation of $f(b')$ may be constructed from these known solutions, namely

$$\bar{f}(b') = \max\{f^k(b') \mid k \in \bar{K}\}$$

Clearly $\bar{f}(b')$ is also a nondecreasing step function and is a lower bound function for $f(b')$, i.e. $\bar{f}(b') \le f(b')$ for all $b' \in \mathbf{R}^m$. The approximation can be improved as new feasible solutions are discovered.

We are interested in a particular "slice" of $f(b')$ and $\bar{f}(b')$: the line segment $\{b + \theta d \mid 0 \le \theta \le 1\}$ where b is the right-hand-side of (P_0) and d is the given direction vector. Define $g(\theta) = f(b + \theta d)$ and $\mathrm{LB}(\theta) = \bar{f}(b + \theta d)$ for $0 \le \theta \le 1$. Then $g(\theta)$ and $\mathrm{LB}(\theta)$ are both step functions and $\mathrm{LB}(\theta) \le g(\theta)$ for all $0 \le \theta \le 1$. If $d \ge 0$, then $g(\theta)$ and $\mathrm{LB}(\theta)$ are both nondecreasing. (See Fig. 1.) There is at least one optimal solution of (PIP) associated with each step of $g(\theta)$. Solving (PIP) is equivalent to constructing $g(\theta)$ by finding at least one x solution for each of its steps.

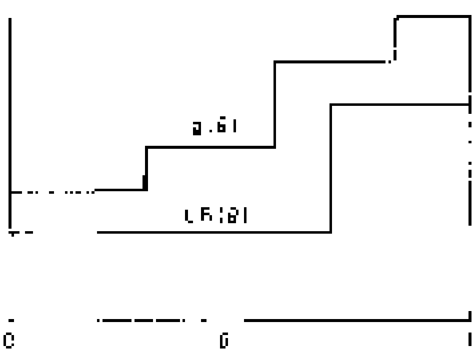


Fig. 1. Typical $g(\theta)$ and $\mathrm{LB}(\theta)$ functions.

The procedure for constructing $\mathrm{LB}(\theta)$ from the known feasible solutions is as follows. For each $k \in \bar{K}$ define

$$\theta_1^k = \min\left\{\theta \,\middle|\, \sum_{j=1}^{n} A_j x_j^k \leq b + \theta d\right\} \tag{3.1}$$

$$\theta_2^k = \max\left\{\theta \,\middle|\, \sum_{j=1}^{n} A_j x_j^k \leq b + \theta d\right\} \tag{3.2}$$

where $\theta_1^k = \theta_2^k = +\infty$ if the indicated set is empty. Then

$$\mathrm{LB}^k(\theta) = \begin{cases} \sum_{j=1}^{n} r_j x_j^k & \text{if } \theta_1^k \leq \theta \leq \theta_2^k, \\ -\infty & \text{otherwise,} \end{cases} \tag{3.3}$$

$$\mathrm{LB}(\theta) = \max\{\mathrm{LB}^k(\theta) \mid k \in \bar{K}\}. \tag{3.4}$$

The solutions which determine $\mathrm{LB}(\theta)$ will be called the incumbents. Each one is incumbent for a particular interval of θ.

4. The upper bound functions

Consider a given partial solution x^q. In order to demonstrate that no descendant of x^q could be optimal for any (P_θ), we need an upper bound on the return achieved by any descendant and this upper bound must depend on θ. Such an upper bound can be obtained by introducing (θd) into the relaxed candidate problem (CP_q^R). Define

$$\mathrm{UB}^q(\theta) = \sum_{j \in S^q} r_j x_j^q + \max \sum_{j \in F^q} r_j x_j$$

$$\text{subject to } \sum_{j \in F^q} a_{ij} x_j \leq b_i + \theta d_i - \beta_i^q, \quad 1 \leq i \leq m,$$

$$0 \leq x_j \leq 1, \quad j \in F^q,$$

so that $\mathrm{UB}^q(0) = \mathrm{UB}^q$. It is well known that $\mathrm{UB}^q(\theta)$ is concave and piecewise linear on $0 \leq \theta \leq 1$. The function $\mathrm{UB}^q(\theta)$ could be obtained explicitly by ordinary parametric linear programming. The computational burden involved in doing this for every candidate problem could be quite substantial, however. Fortunately any dual feasible solution of (CP_q^R) can be used to construct a linear upper bound function for $\mathrm{UB}^q(\theta)$. An optimal dual solution of (CP_q^R), barring degeneracy, yields the first linear segment of $\mathrm{UB}^q(\theta)$. By linear programming duality we know that:

$$\mathrm{UB}^q(\theta) = \sum_{j \in F^q} c_j x_j^q + \min \sum_{i=1}^{m} u_i(b_i + \theta d_i - \beta_i^q) + \sum_{j=1}^{n} v_j$$

$$\text{subject to} \sum_{i=1}^{m} u_i a_{ij} + v_j \geq c_j, \quad j \in J^q,$$

$$u_i \geq 0, \quad 1 \leq i \leq m,$$

$$v_j \geq 0, \quad 1 \leq j \leq n.$$

For notational convenience we have included all of the v_j variables, even though $v_j = 0$ for $j \in F^q$ in any optimal solution. Let D^q denote the dual feasible region

$$D^q = \left\{ (u, v) \geq 0 \;\middle|\; \sum_{i=1}^{m} u_i a_{ij} + v_j \geq c_j \quad \text{for } j \in J^q \right\}.$$

Since the primal variables are all bounded and at least one (P_θ) is feasible, we may conclude that D^q is non-empty. Let $\{(u^t, v^t) \mid t \in T^q\}$ and $\{(y^s, z^s) \mid s \in S^q\}$ denote the sets of extreme points and extreme rays, respectively, of D^q. Taking $e = (1, \ldots, 1)$ we have

$$\mathrm{UB}^q(\theta) \leq \sum_{j \in F^q} c_j x_j^q + u^t(b + \theta d - \beta^q) + v^t e$$

for all $t \in T^q$, with equality if (u^t, v^t) is optimal for the objective function $u(b + \theta d - \beta^q) + ve$. As a function of θ then, the return achieved by any descendant of x^q is bounded above by:

$$\mathrm{UB}^q(\theta; t) = (u^t d)\theta + \left[\sum_{j \in F^q} c_j x_j^q + u^t(b - \beta^q) + v^t e \right]$$

for any $t \in T^q$. This is a linear function of θ and, since $u^t \geq 0$, it is nondecreasing if $d \geq 0$.

In the modified branch-and-bound algorithm for (PIP), linear programming is applied to (CP_0^q) as usual. The functions $\mathrm{UB}^q(\theta; t)$ are obtained at no extra cost. The function obtained from an optimal dual solution will be denoted $\mathrm{UB}^q(\theta; *)$. Barring degeneracy, $\mathrm{UB}^q(\theta; *)$ coincides with the first linear segment of $\mathrm{UB}^q(\theta)$ (see Fig. 2). As in conventional branch and bound, if the dual simplex method is used, then suboptimal dual solutions can be used to perform additional weaker tests.

If (CP_0^q) is infeasible, then the simplex method will terminate with an extreme point $(u^t, v^t) \geq 0$ and an extreme ray $(y^s, z^s) \geq 0$, such that

$$y^s(b - \beta^q) + z^s e < 0.$$

If $y^s d \leq 0$, then we may conclude that $\mathrm{UB}^q(\theta) = -\infty$ for all $0 \leq \theta \leq 1$. If $y^s d > 0$, then $\mathrm{UB}^q(\theta) = -\infty$ for $0 \leq \theta < \theta^s$ and $\mathrm{UB}^q(\theta) \leq \mathrm{UB}^q(\theta; t)$ for $\theta^s \leq \theta \leq 1$, where

$$\theta^s = \frac{-y^s(b - \beta^q) - z^s e}{y^s d}.$$

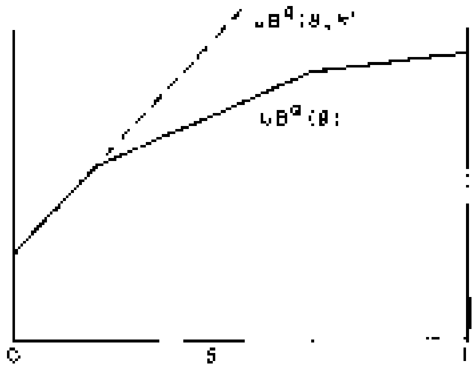


Fig. 2. Typical $UB^q(\theta)$ and $UB^q(\theta, \pi)$ functions.

5. A branch-and-bound algorithm for (PIP)

Now that the upper and lower bound functions have been derived, the generalized bounding test may be stated. The partial solution x^q does not have a descendant that is better than an incumbent if

$$UB^q(\theta) \le LB(\theta), \quad \text{for all } 0 \le \theta \le 1,$$

or if

$$UB^q(\theta, \pi) \le LB(\theta), \quad \text{for all } 0 \le \theta \le 1,$$

for some $\pi \in T^q$. This test is the basis for a modified branch-and-bound algorithm that can solve (PIP).

Step 1. Place (P_0) in the candidate list and set $LB(\theta) = -\infty$ for $0 \le \theta \le 1$.

Step 2. If the candidate list is empty, stop. $LB(\theta) = g(\theta)$ for $0 \le \theta \le 1$.

Step 3. Select a candidate problem and remove it from the candidate list. Call it (CP^q).

Step 4. Solve (CP^q_0). If it is infeasible, obtain the appropriate dual extreme point (u^q, v^q) and extreme ray (y^q, z^q). Otherwise obtain an optimal dual solution (u^q, v^q).

Step 5. I. (CP^q_0) infeasible.

(a) $y^q d \le 0$. Go to Step 2.

(b) $y^q d > 0$. Set $\theta^* = [-y^q(b - \beta) - z^q e]/y^q d$. If $UB^q(\theta; \pi) \le LB(\theta)$ for all $\theta^* \le \theta \le 1$, go to Step 2.

II. (CP^q_0) feasible. If $UB^q(\theta; \pi) \le LB(\theta)$ for all $0 \le \theta \le 1$, go to Step 2.

Step 6. If the optimal primal solution of (CP^q_0) is all integer, use it to update $LB(\theta)$.

Step 7. Use a heuristic to find feasible solutions of (CP^q) with right-hand-side $(b + \theta d)$ for several values of θ. Use these feasible solutions to update $LB(\theta)$.

Step 8. Separate (CP^k) into two new candidate problems $(\mathrm{CP}^{k'})$ and $(\mathrm{CP}^{k''})$ by choosing $p \in J^k$ and setting $F^{k'} = F^{k''} = F^k \cup \{p\}$, $x_p^{k'} = 0$, $x_p^{k''} = 1$. Place $(\mathrm{CP}^{k'})$ and $(\mathrm{CP}^{k''})$ in the candidate list and return to Step 2.

The validity of the generalized bounding test insures that an optimal solution for every (P_θ), $0 \le \theta \le 1$, will be found by the search. At worst, an optimal solution may not be discovered until the bottom of the branch-and-bound tree is reached ($F^k = J$). This guarantees that $\mathrm{LB}(\theta)$ will coincide with $g(\theta)$ by the time the algorithm is finished. It remains only to show how the optimal solutions are identified.

Let $\{x^k \mid k \in \bar{K}\}$ be the set of incumbents when the algorithm terminates. Let $\theta \in [0, 1]$ and suppose that (P_θ) is feasible, $g(\theta) > -\infty$. From the construction of $\mathrm{LB}(\theta)$, (3.1)–(3.4), we know that there is some $k \in \bar{K}$ such that

$$
\begin{aligned}
g(\theta) &= \mathrm{LB}(\theta) \\
&= \mathrm{LB}^k(\theta) \\
&= \sum_{j=1}^{n} r_j x_j^k > -\infty.
\end{aligned}
$$

Furthermore, $\mathrm{LB}^k(\theta) > -\infty$ means that $\theta_1^k \le \theta \le \theta_2^k$, or equivalently that

$$\sum_{j=1}^{n} A_j x_j^k \le b - \theta d.$$

Since x^k is feasible for (P_θ) and its return is equal to $g(\theta)$, it follows that x^k is optimal for (P_θ). To summarize, if $k \in \bar{K}$ and $\theta \in [0, 1]$, then x^k is optimal for (P_θ) if and only if

(i) $\sum_{j=1}^{n} A_j x_j^k \le b - \theta d$

(ii) $\sum_{j=1}^{n} r_j x_j^k = g(\theta)$.

At Step 6, in contrast to the prototype algorithm, x^k is not fathomed when the optimal primal solution of (CP_0^k) is all integer. This is because x^k may have other descendants which are optimal for $\theta > 0$. The use of heuristics at Step 7, while in principle optional, is an important part of the algorithm since integer solutions of (CP_0^k) can only yield $\mathrm{LB}(\theta) = \mathrm{LB}(0)$ for $0 \le \theta \le 1$. The heuristics are needed to produce stronger values of $\mathrm{LB}(\theta)$ for $\theta > 0$.

As with the prototype algorithm, the above procedure will admit considerable variation and refinement. If the dual simplex method is used, then suboptimal dual solutions can be used to perform additional bounding tests. Cutting planes can be generated for any candidate problem to give stronger upper bound functions. Parametric linear programming can be used to generate more than the first segment of $\mathrm{UB}^k(\theta)$. If a candidate problem with an all-integer LP solution has to be separated at Step 8, then the same LP solution is optimal for one of the two new candidates and does not have to be recomputed. Extensive experimentation will be required to determine the most effective computational tactics.

6. Example

In this section the algorithm will be applied to a simple example. In order to illustrate all of the different cases that can arise, the parameterization will be done over a relatively large interval. The test problem is

$$\max \; 10x_1 + 15x_2 - 10x_3 + 5x_4$$

$$\text{subject to } \; 2x_1 - 5x_2 + 5x_3 + 1x_4 \leq 4 + \theta 4$$

$$4x_1 + 2x_2 + 1x_3 + 1x_4 \leq 4 + \theta 4$$

$$x_j \in \{0, 1\} \quad 1 \leq j \leq 4$$

Thus $b = (4, 4)$, $d = (4, 4)$ and increasing θ from zero to one amounts to doubling the right-hand-side. A picture of the optimal return function $f(b')$ is given in Fig. 3.

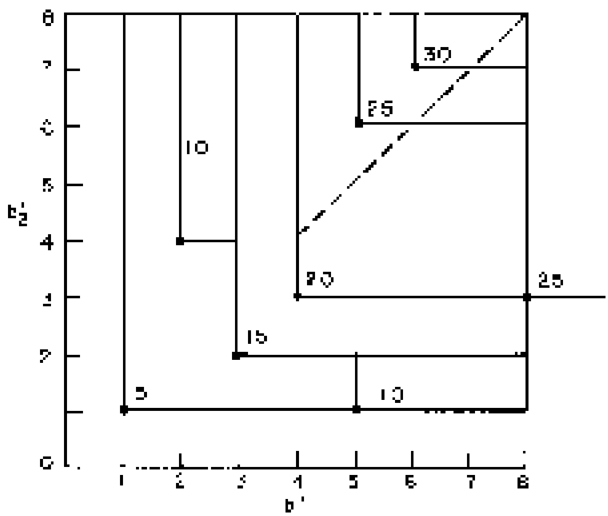


Fig. 3. The optimal return function $f(b')$.

The dashed line indicates the line segment of interest: $\{b + \theta d \mid 0 \leq \theta \leq 1\}$. It is clear from this picture that three optimal solutions must be found, with values of 20, 25, and 30. These solutions are (0, 1, 0, 1), (1, 1, 0, 0), and (1, 1, 0, 1) respectively. The $g(\theta)$ function, shown in Fig. 4, is

$$g(\theta) = \begin{cases} 20 & \text{for } 0 \leq \theta < 1/2, \\ 25 & \text{for } 1/2 \leq \theta < 3/4, \\ 30 & \text{for } 3/4 \leq \theta \leq 1. \end{cases}$$

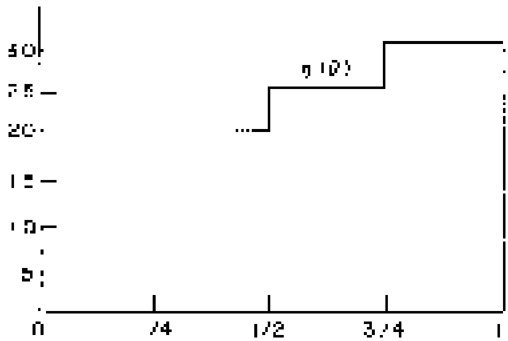


Fig. 1. The parametric function $g(\theta)$.

The optimal LP solution of (P_2) is $x = (1/2, 1, 0, 0)$, $u = (5, 0)$, $v = (0)$ with value $UB'' = 20$. The rounded down solution has value 15 and is feasible for $\theta \geq 0$; the rounded up solution has value 25 and is feasible for $\theta \geq 1/2$. This provides an initial lower bound function:

$$\mathrm{LB}(\theta) = \begin{cases} 15 & \text{for } 0 \leq \theta < 1/2, \\ 25 & \text{for } 1/2 \leq \theta \leq 1. \end{cases}$$

The complete branch-and-bound tree is displayed in Fig. 3. The nodes will be discussed in the order in which they were created.

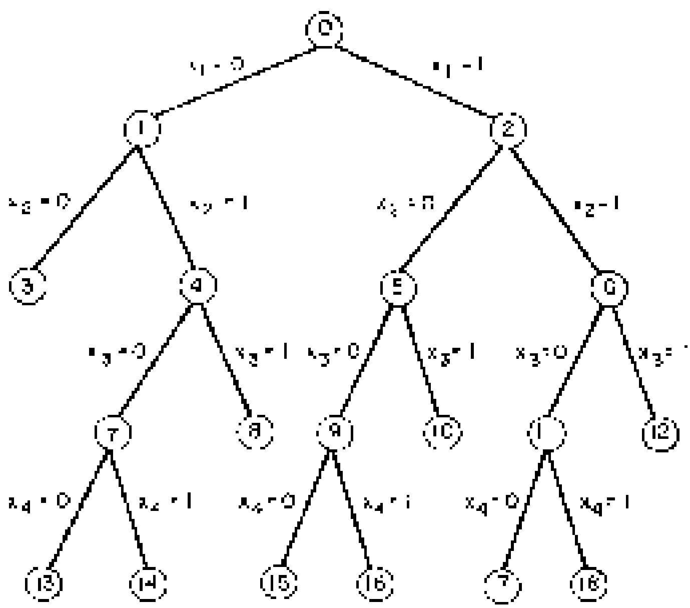


Fig. 3. Branch-and-bound tree for the example.

Node 1. LP solution: $x = (0,1,0,1)$, $u = (5,0)$, $v = (0)$, $\mathrm{UB}^1 = 20$, $\mathrm{UB}^1(\theta;x) = 20\theta + 20$. The LP solution is all integer and is feasible for $\theta \geq 0$. Therefore the lower bound function may be improved:

$$\mathrm{LB}(\theta) = \begin{cases} 20 & \text{for } 0 \leq \theta < 1/2, \\ 25 & \text{for } 1/2 \leq \theta \leq 1. \end{cases}$$

The bounding test for node 1 is shown in Fig. 6. Node 1 is not fathomed.

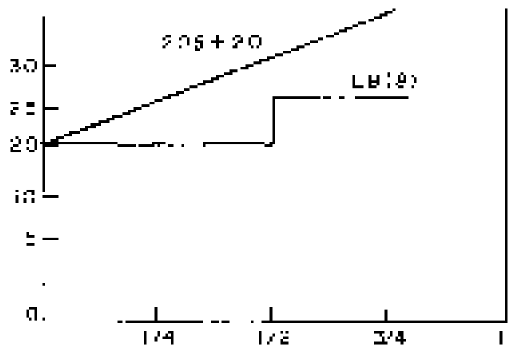


Fig. 6. Bounding test for node 1.

Node 2. LP solution: $x = (1,0,0,0)$, $u = (0,10)$, $v = (0)$, $\mathrm{UB}^2 = 10$, $\mathrm{UB}^2(\theta;x) = 40\theta + 10$. The bounding test, shown in Fig. 7, is not successful. Notice that if we were only interested in solving (P_0) we would be finished. Node 1 has an all integer solution with value 20 and node 2 has upper bound $\mathrm{UB}^2 = 10 < 20 = \mathrm{LB}(0)$.

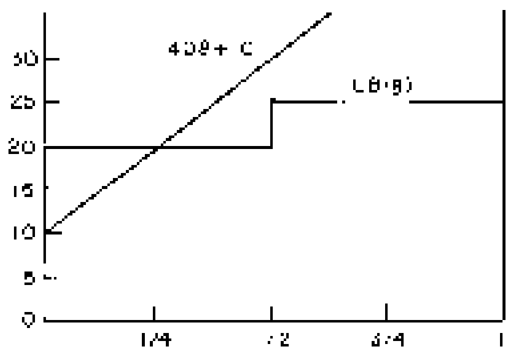


Fig. 7. Bounding test for node 2.

Node 3. LP solution: $x = (0,0,3/5,1)$, $u = (2,0)$, $v = (0,0,0.3)$, $\mathrm{UB}^3 = 11$, $\mathrm{UB}^3(\theta;x) = 8\theta + 11$. The bounding test, shown in Fig. 8, is successful and node 3 is fathomed.

Node 4. Same as node 1, since optimal LP solution at node 1 has $x_2 = 1$.

Node 5. Same as node 2, since optimal LP solution at node 2 has $x_2 = 0$.

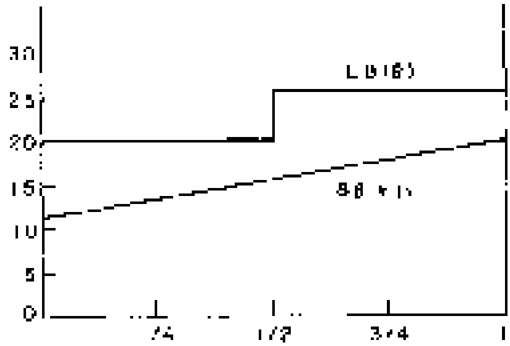


Fig. 8. Bounding test for node 3.

Node 6. LP is infeasible. The dual extreme point is $u = (0, 10)$, $v = (0)$ and the extreme ray is $y = (0, 1)$, $z = (0)$. The critical value of θ is $(-y(b-\beta^6)-ze)/yd = 1/2$. Thus $UB^6(\theta) = -\infty$ for $0 \le \theta < 1/2$ and $UB^6(\theta;*) = 40\theta + 5$ for $1/2 \le \theta \le 1$. The bounding test is shown in Fig. 9.

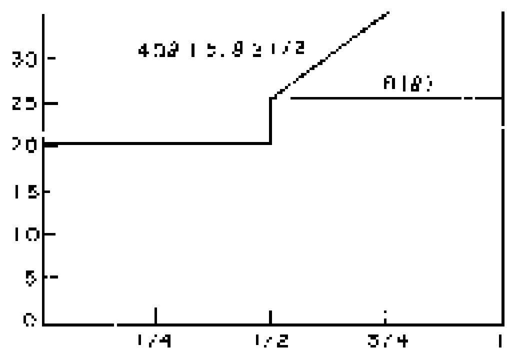


Fig. 9. Bounding test for node 6.

Node 7. Same as nodes 1 and 4, since optimal LP solution for node 4 has $x_5 = 0$.

Node 8. LP is infeasible. The dual extreme point is $u = (5, 0)$, $v = (0)$ and the extreme ray is $y = (1, 0)$, $z = (0)$. The critical value of θ is $(-y(b-\beta^8)-ze)/yd = 1$, so $UB^8(\theta) = -\infty$ on $0 \le \theta < 1$ and node 8 is fathomed.

Node 9. Same as nodes 2 and 5, since optimal LP solution for node 5 has $x_5 = 0$.

Node 10. LP is infeasible. The dual extreme point is $u = (5, 40)$, $v = (0)$ and the extreme ray is $y = (1, 0)$, $z = (0)$. The critical value of θ is $(-y(b-\beta^{10})-ze)/yd = 3/4$. Thus $UB^{10}(\theta) = -\infty$ for $0 \le \theta < 3/4$ and $UB^{10}(\theta;*) = 20\theta + 5$ for $3/4 \le \theta \le 1$. Node 10 is therefore fathomed. See Fig. 10.

Node 11. LP is infeasible. The dual extreme point is $u = (0, 5)$, $v = (0)$ and the extreme ray is $y = (0, 1)$, $z = (0)$. The critical value of θ is $(-y(b-\beta^{11})-ze)/yd =$

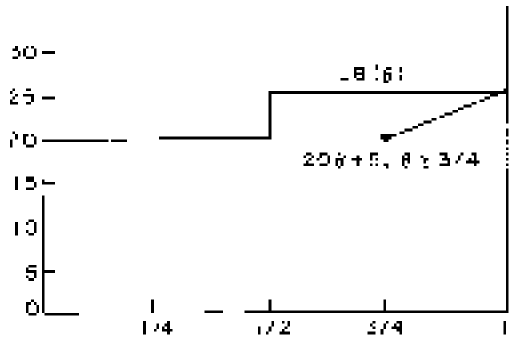


Fig. 10. Bounding test for node 10.

1/2. Thus $\mathrm{UB}^{11}(\theta) = \infty$ for $0 \leq \theta < 1/2$ and $\mathrm{UB}^{11}(\theta; *) = 20\theta + 15$ for $1/2 \leq \theta \leq 1$. Node 11 is not fathomed. See Fig. 11.

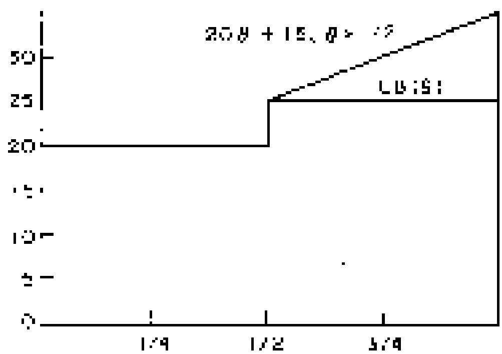


Fig. 11. Bounding test for node 11.

Node 12. LP is infeasible. The dual extreme point is $u = (5,0)$, $v = (0)$ and the extreme ray is $y = (1,0)$, $z = (0)$. The critical value of θ is $(-y(b - B^{12}) - ze)/yd = 1$. Therefore $\mathrm{LB}^{12}(\theta) = -\infty$ for $0 \leq \theta \leq 1$ and node 12 is fathomed.

Nodes 13–18 are all at the bottom level of the search tree. The solution for node 18, $(1, 1, 0, 1)$, has value 30 and is feasible for $\theta \geq 3/4$. The lower bound function may be improved by redefining $\mathrm{LB}(\theta) = 30$ for $3/4 \leq \theta \leq 1$. $\mathrm{LB}(\theta)$ now coincides with $g(\theta)$ on $0 \leq \theta \leq 1$. The algorithm terminates since the candidate list is empty.

The amount of extra computation required to solve (PIP), as compared to (P_θ), depends on the length of the interval of parameterization. When this interval is small, the burden imposed by parameterization may be slight or even negligible. When it is large, however, as illustrated in this example, the burden can be quite substantial.

7. Computational results

The ideas presented above were tested by incorporating them into a branch-and-bound computer code [3]. The results for four test problems are presented in Table 1. In each run the direction vector d was taken as some percentage of the right-hand-side b. For example, if $d = 5\% b$ then (PIP) has right-hand-sides $b + \theta(.05)b$ for $0 \leq \theta \leq 1$. The column headed "solutions" gives the number of optimal solutions found, or equivalently the number of steps of the $g(\theta)$ function. "Heuristic" is the number of (evenly spaced) θ values for which the heuristic is applied at Step 7. The problems are of the capital budgeting type and the heuristic employed is that of Toyoda [12]. "Pivots" is the total number of linear programming pivots and "time" is the total solution time in seconds on an IBM 370/168.

Table 1. Computational results for four test problems.

m	n	d	solutions	heuristic	pivots	time
5	15	0	1	1	39	0.239
		0.05b	4	10	65	0.547
		0.10b	5	10	95	0.615
		0.15b	7	10	124	1.091
		0.20b	8	10	130	1.170
		0.25b	10	10	171	1.534
		0.50b	16	20	315	3.162
5	30	0	1	1	153	1.605
		0.05b	11	10	529	5.114
		0.10b	28	20	1175	18.615
		0.15b	37	20	2066	43.304
10	28	0	1	1	65	1.155
		0.05b	16	10	175	4.489
		0.10b	29	20	645	15.129
		0.15b	42	20	1621	30.998
30	50	0	1	1	180	5.742
		0.025b	6	5	481	9.486
		0.05b	12	10	1390	32.185

These results illustrate quite clearly how the computational burden increases as the interval of parameterization is lengthened. In order to facilitate comparison with our results by other researchers we have included the data for the 5×30 problem as Table 2 and the corresponding $g(\theta)$ function for a 15% increase in b as Table 3.

Table 2. The 5 × 30 test problem

a_{1j}	a_{2j}	a_{3j}	a_{4j}	a_{5j}	c_j
188	93	31	86	164	936
92	179	99	57	96	695
6	140	96	42	2	391
90	155	95	90	176	1152
91	102	84	101	140	980
44	112	136	3	106	1000
109	126	106	101	88	815
106	21	23	34	68	109
171	39	20	25	84	807
64	67	124	72	131	156
97	29	42	96	57	548
35	55	58	36	15	335
51	72	43	3	17	316
98	17	43	88	4	528
86	11	44	97	47	36
70	42	2	77	45	373
27	15	68	50	11	38
91	61	55	14	77	3
68	51	66	77	36	600
15	30	22	88	49	392
12.2	2.8	6.8	11.3	2.9	92
15.1	15.0	8.3	13.8	11.7	4
3.3	2.6	8.9	4.5	19.2	29
7.4	1.5	3.1	17.1	16.1	61
7.0	17.0	16.5	11.8	3.8	2
1.2	3.5	2.3	17.1	18.0	40
7.0	5.1	9.7	19.1	6.8	17
17.0	16.3	4.7	5.0	9.0	16
12.8	13.2	1.8	18.2	16.9	50
9.4	13.9	11.0	3.6	13.8	118
b = 600	600	500	700	800	

Table 3. The $g(\theta)$ function for a 15% increase in b; 5 × 30 problem.

θ	$g(\theta)$	θ	$g(\theta)$	θ	$g(\theta)$
0.0	7515	0.41573	7846	0.60166	8112
0.01835	7578	0.42800	7865	0.62333	8141
0.05524	7607	0.43714	7891	0.71083	8161
0.10286	7612	0.45667	7910	0.73333	8171
0.11250	7630	0.45809	7931	0.75809	8181
0.11416	7696	0.47633	7942	0.77500	8204
0.14584	7725	0.49428	7947	0.79333	8224
0.20952	7777	0.49809	7957	0.82083	8253
0.25738	7806	0.51809	7991	0.87916	8270
0.30666	7807	0.54198	8008	0.93583	8283
0.32750	7822	0.56095	8023	0.99416	8300
0.34953	7836	0.56265	8049		
0.37499	7839	0.59416	8060		

References

[1] V.J. Bowman, The structure of integer programs under the Hermitian normal form, *Operations Res.* 22 (1974) 1067–1080.

[2] A.M. Geoffrion and R.E. Marsten, Integer programming algorithms: A framework and state-of-the-art-survey, Management Sci. 18 (1972) 465–491.

[3] R.E. Marsten and T.L. Morin, A hybrid approach to discrete mathematical programming, Sloan School of Management, MIT, Cambridge, Mass., July, 1975.

[4] T.L. Morin and R.E. Marsten, An algorithm for nonlinear knapsack problems, Management Sci. 22 (1976) 1147–1158.

[5] T.L. Morin and R.E. Marsten, Branch and bound strategies for dynamic programming, Operations Research 24 (4) (1976) 611–627.

[6] R.M. Nauss, Parametric integer programming, Ph.D. Dissertation, University of California, Los Angeles, January, 1975.

[7] H. Noltemeier, Sensitivitätsanalyse bei diskreten linearen Optimierungsproblemen, in M. Beckmann and H.P. Künzi, eds., *Lecture Notes in Operations Research and Mathematical Systems*, # 30. (Springer, Berlin, 1970).

[8] C.J. Piper and A.A. Zoltners, Implicit enumeration based algorithms for postoptimizing zero-one problems, Management Sciences Research Report, No. 313, Graduate School of Industrial Administration, Carnegie-Mellon University, March, 1973.

[9] C.J. Piper and A.A. Zoltners, Some easy postoptimality analysis for zero-one programming, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa., 1975 (forthcoming in *Management Science*).

[10] G.M. Roodman, Postoptimality analysis in zero-one programming by implicit enumeration, Naval Research Logistics Quarterly, Vol. 19, 1972, pp. 435–447.

[11] G.M. Roodman, Postoptimality analysis in integer programming by implicit enumeration: The mixed integer case, The Amos Tuck School of Business Administration, Dartmouth College, October, 1973.

[12] Y. Toyoda, A simplified algorithm for obtaining approximate solutions to zero-one programming problems, *Management Sci.* 21 (1975) 1417–1427.

Annals of Discrete Mathematics 1 (1977) 391–393

AN EXAMPLE OF DUAL POLYTOPES IN THE UNIT HYPERCUBE

J.F. MAURRAS
Département Méthodes d'Optimisation, Electricité de France, 92140 Clamart, France

Let $N = \{1, 2, \ldots, n\}$ and let S be a subset of N with $S = \{s_1, s_2, \ldots, s_k\}$ such that

$$0 \le s_1 < s_2 < \cdots < s_k \le n.$$

Let P be the polytope in R^n defined by the system of inequalities:

$$0 \le x_j \le 1, \; j = 1, \ldots, n;$$

$$s_1 \le \sum_{j=1}^{n} x_j \le s_k;$$

for each $I \subset N$ such that $s_i < |I| < s_{i+1}$

$$\sum_{j \in I} x_j - \frac{k_i}{s_{i+1} - s_i - k_i} \sum_{j \notin I} x_j \le s_i,$$

where $k_i = |I| - s_i$.

Theorem 1 [1]. *P is the convex hull of the zero-one vectors $x_j = 0$ or 1, $j = 1, \ldots, n$, such that*

$$\sum_{j=1}^{n} x_j \in S.$$

Theorem 2 [1]. *The facets of P are given by:*
$x_j \le 1$, $j = 1, \ldots, n$, *unless S is one of* $\{0, n\}$, $\{0, 1, n\}$, $\{1, n\}$,
$x_j \ge 0$, $j = 1, \ldots, n$, *unless S is one of* $\{0, n\}$, $\{0, n-1, n\}$, $\{0, n-1\}$;
$\sum x_j \ge s_1$ *if and only if* $s_1 > 0$;
$\sum x_j \le s_k$ *if and only if* $s_k < n$;

$$\sum_{j \in I} x_j - \frac{k_i}{s_{i+1} - s_i - k_i} \sum_{j \notin I} x_j \le s_i$$

for $s_i < |I| < s_{i+1}$, *if and only if*
(i) $0 < s_i$ *and* $s_{i+1} < n$, *or*
(ii) $s_i = 0$ *and* $s_{i+1} < n$ *and* $|I| = 1$, *or*
(iii) $s_i > 0$ *and* $s_{i+1} = n$ *and* $|I| = n - 1$.

These polytopes are invariant under permutations of the indices of the variables. We shall investigate here the polytopes of this class which are duals (i.e., their face lattices are anti-isomorphic).

Theorem 3. *For $n \ge 3$, except for the n-simplices, which are duals and self-duals, the only dual polytopes in this class are those defined by the following sets:*

$\{1, n-1\}$ *and* $\{0, 1, \dots, n\}$;

$\{0, 1, n-1\}$ *and* $\{1, 2, \dots, n\}$;

$\{1, n-1, n\}$ *and* $\{0, 2, \dots, n-1\}$;

$\{0, 1, n-1, n\}$ *and* $\{1, \dots, n-1\}$.

Sketch of proof. The first dual pair is proven by showing that the dual of the unit hypercube, i.e. P and $S = \{0, 1, \dots, n\}$, is given by P when $S = \{1, n-1\}$. This can be proven from the result, using the terminology of [2], that the n cross polytope (i.e., the convex hull of $(0, \dots, \pm 1, 0, \dots, 0)$, for all i) is the dual of the unit hypercube. The n cross polytope is facially equivalent to P for $S = \{1, n-1\}$; this P is the convex hull of $(0, \dots, +1, 0, \dots, 0)$, $(1, \dots, 1, 0, 1, \dots, 1)$ for all i. In fact, there is an affine transformation which takes this P to the n cross polytope.

A different proof of this first pair is the heart of the proof of the entire theorem. A mapping from facets of the first P to the vertices of the unit hypercube is:

$$\sum_{i=1}^{n} x_i \ge 1 \to (0, \dots, 0); \tag{1}$$

$$x_i \ge 0 \to (0, \dots, \overset{i}{1}, 0, \dots, 0); \tag{2}$$

$$\sum_{i \in I} x_i - \frac{|I|-1}{n-|I|-1} \sum_{j \notin I} x_j \ge 1, \quad 1 < |I| < n-1, \tag{3}$$

$$\to x,\ x_j = 0 \text{ for } j \in I,\ x_j = 1 \text{ for } j \notin I;$$

$$x_i \le 1 \to (1, \dots, \overset{i}{0}, 1, \dots, 1); \tag{4}$$

$$\sum_{i=1}^{n} x_i \le n-1 \to (1, \dots, 1). \tag{5}$$

The proof of the "only if" part of the theorem requires showing, using invariance of P, that the pairs given with a mapping much as the one above, are the only possible dual pairs.

References

[1] P. Camion and J.F. Maurras, Polyhedrons with Vertices in $[0,1]^n$, submitted for publication.
[2] P. McMullen and G.C. Shephard, *Convex Polytopes and the Upper Bound Conjecture*, London Math Society Lecture Notes Ser. No. 3 (Cambridge University Press, London, 1971).

Annals of Discrete Mathematics 1 (1977) 395–402

IMPLICIT ENUMERATION WITH GENERALIZED UPPER BOUNDS

P. MEVERT and U. SUHL
Department of Operations Research, Free University of Berlin, D-1000 Berlin 33, Germany

A number of planning problems can be formulated as (0–1)-programs where all variables can be grouped into special ordered sets or generalized upper bounds.

An implicit enumeration algorithm was developed and implemented for this class of problems. The generalized upper bounds are handled implicitly. Only non zero elements of the large but sparse constraint matrix are stored explicitly and chained row-wise and column-wise. The storage structure allows for very efficient testing of partial solutions. Preliminary numerical results indicate that even large-scale problems can be solved efficiently.

1. Introduction

Balas introduced the concept of implicit enumeration more than ten years ago [3]. Since then, a number of major improvements have been suggested, e.g. [4, 5, 10, 13, 14, 15, 16, 20, 23, 27]. Numerical results, however, have been somewhat disappointing and success has been limited to problems of small or moderate size, in general.

By contrast, quite large problems were solved successfully using LP-based branch-and-bound codes. This may have led to the belief that these methods are, in general, more powerful than implicit enumeration.

It should be pointed out, however, that many man-years have gone into the development of LP-based codes like UMPIRE, MPSX-MIP, or APEX. On the other hand, very little effort has been spent, to our knowledge, to develop comparable implicit enumeration codes. Most authors report that their results were obtained using an experimental code on small artificial test problems.

An important fact that seems to have been overlooked is the fact that realistic problems differ from the usual small test problems in two essential aspects. Firstly, they exhibit in most cases special structure. Secondly, the matrix of coefficients is always sparse. Exploiting these two characteristics of real problems and being careful to minimize data handling in the implementation may increase the efficiency of implicit enumeration codes to the extent that even large-scale problems can be solved successfully. The following is an example of this approach.

We consider (0–1) problems where the set of all variables can be partitioned into subsets and exactly one variable from each subset must take on the value one. We will use the terminology special ordered sets, convexity constraints, multiple choice constraints, and generalized upper bounds interchangeably. (See [6] for the original more general concept of special ordered sets.)

This class of problems contains a large number of applications, including assembly line balancing [18], resource constrained network scheduling [2], distribution problems [9], time-table problems [19], and a number of other scheduling problems [2, 7, 8]. Certain production planning problems with setup costs and location-distribution problems can be formulated as mixed integer programming problems and solved by Benders' Method [12]. In these and other cases, the master problem exhibits a structure such that all variables can be grouped into special ordered sets, as defined above.

Thus, we consider the following problem P:

minimize z

$$\text{s.t.}\quad z = \sum_{j\in J} c_j x_j, \tag{0}$$

$$\sum_{j\in J} a_{ij} x_j \geq b_i, \quad \forall i \in M, \tag{1}$$

$$\sum_{j\in J_k} x_j = 1, \quad \forall k \in K, \tag{2}$$

$$x_j = 0 \quad \text{or } 1, \quad \forall j \in J, \tag{3}$$

where J_k are the special ordered sets with

$$\bigcup_{k\in K} J_k = J \quad \text{and} \quad J_i \cap J_k = \emptyset \quad \text{for } i \neq k.$$

Without loss of generality, we assume that

$$c_j \geq 0 \quad \text{for all } j \in J.$$

Following standard terminology we define a partial solution as a projection of the solution space onto a lower dimensional space by assigning binary values to a subset $S \subset J$ of the variables x_j. An admissible partial solution is an assignment of binary values to the variables x_j, $j \in S \subset J$ such that each special ordered set contains at most one variable with value 1. We define:

S index set of variables to which binary values are assigned;

S_1 index set of variables assigned the value 1;

S_0 index set of variables assigned the value 0;

$b_i(S) = b_i - \sum_{j\in S_1} a_{ij}$ current right-hand-side;

$z(S) = \sum_{j\in S_1} c_j$ current value of the partial solution S;

$V_1(S) = \{i \in M \mid h_i(S) > 0\}$ index set of general constraints (1) which would be violated if the partial solution were completed by assigning $x_j = 0$ to all $j \in J - S$;

$V_2(S) = \{k \in K \mid S_1 \cap J_k = \emptyset\}$ index set of convexity constraints (2) which would be violated if S were completed by $x_j = 0$, $j \in J - S$;

$F(S)$ index set of free (unassigned) variables x_j, $j \in J - S$;

$$L_k(S) = \begin{cases} J_k \cap F(S) & \text{if } S_1 \cap J_k = \emptyset, \\ \emptyset & \text{otherwise,} \end{cases}$$

index set of admissible variables from the special ordered set J_k;

$L(S) = \bigcup_{k \in K} L_k(S)$ index set of admissible variables.

Note that a free variable is called admissible if no other variable in the same special ordered set is assigned the value 1.

Then each admissible partial solution defines a subproblem $P(S)$:

$$\min \sum_{j \in L(S)} c_j x_j + z(S), \tag{0'}$$

$$\sum_{j \in L(S)} a_{ij} x_j \geq b_i(S), \quad i \in M, \tag{1'}$$

$$\sum_{j \in L_k(S)} x_j = 1, \quad k \in V_2(S), \tag{2'}$$

$$x_j = 0 \quad \text{or } 1, \quad j \in L(S). \tag{3'}$$

Any feasible solution of $P(S)$ defines a feasible completion of S by assigning the value 0 to all remaining free variables. A minimal completion corresponds to a minimal solution of $P(S)$.

Note that $P(S)$ is, in general, a much smaller problem than P, since $L(S) \subset F(S) \subset J$. This fact will be used in the subsequent algorithm. Further, if $V_1(S) = V_2(S) = \emptyset$ (i.e. all $h_i(S) \leq 0$ and all convexity constraints are satisfied) then $x_j = 0$, $j \in J - S$ is a minimal completion of S.

A partial solution S is said to be fathomed if

(a) it can be shown that the minimal value of $P(S)$ is not less than z_{best}, where z_{best} is the value of the best feasible solution to P, found so far, or

(b) it can be determined that $P(S)$ has no solution; or

(c) $P(S)$ is solved, i.e. all completions of S have been (implicitly) enumerated.

A variable x_j, $j \in S$ is said to be *fixed* to 1 if the partial solution S with the opposite value $x_j = 0$ instead of $x_j = 1$ has already been fathomed, i.e. is known to possess no completion with a smaller value of the objective function. Similarly, a variable x_j, $j \in S$ is said to be fixed to 0 if the opposite branch $x_j = 1$ has been or need not be investigated.

By contrast, a variable x_j, $j \in S$ is called *set* to a binary value β if the partial solution S with the opposite branch $x_j = 1 - \beta$ instead of $x_j = \beta$ is not excluded from further investigation.

The information on the status of each variable x_j, $j \in J$ is stored in a status vector st which contains information on all free and assigned variables.

In order to keep relevant information on the enumeration history all indices $j \in S$ are stored in chronological order in a partial solution vector $s = (j_1, j_2, \ldots, j_{|S|})$ where j_k is the index of the variable which was set or fixed at the kth level of the enumeration tree. Finally, $V_0(S)$ is conveniently stored in form of a vector v_0 where $v_0(k) = 0$ if $k \in V_0(S)$, and $v_0(k) = 1$ if the kth multiple choice constraint is satisfied.

2. The implicit enumeration procedure

The problem is solved by implicit enumeration using a modification of the procedures suggested in [3], [13], and [15]. The approach is related to [9]. The multiple choice constraints (2) are stored implicitly but the enumeration procedure uses the structure of these constraints explicitly.

The enumeration proceeds in the usual fashion from an admissible partial solution S. An attempt is made to fathom S. If this is successful, the last variable in S which was set to 0 or 1 is replaced by its complement, i.e. the node selection rule is LIFO.

If S cannot be fathomed, one (or more) variables x_j, $j \in L(S)$ are selected to be set or fixed to 0 or 1, depending on the outcome of some tests. Note that only admissible partial solutions can be generated.

The algorithm uses several of the tests which have been suggested in the literature [3, 5, 10, 11, 20]. Only non-zero elements of the constraint matrix (1) are stored and chained row-wise and column-wise, as will be discussed subsequently. In view of the storage structure, tests are preferred which require very little computational or updating effort.

The basic sequence of tests is shown in Fig. 1. It should be mentioned, however, that Fig. 1 is only an approximate description of the algorithm. For example, if a test results in fixing some of the variables, then in some cases the test will be repeated after updating. In order to keep the exposition simple, such details are not given in Fig. 1.

The following steps correspond to the numbers of Fig. 1:

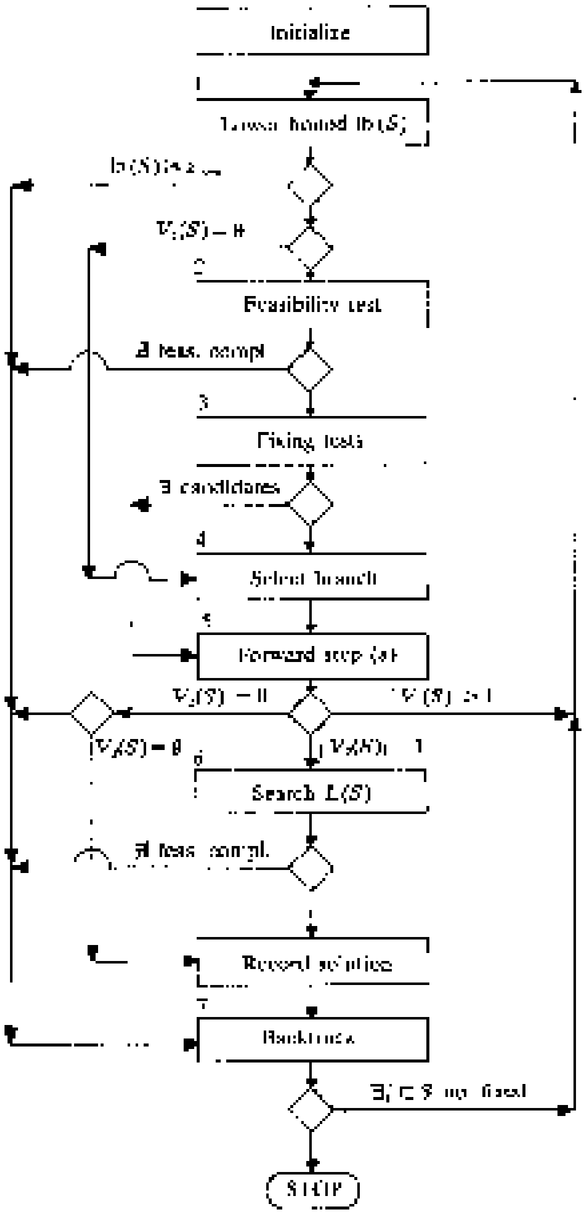


Fig. 1.

(1) Calculate

$$\mathrm{lb}(S) = z(S) + \sum_{i \in V_1(S)} \min_{j \in L_i(S)} \{c_j\},$$

where $\mathrm{lb}(S)$ is a lower bound for $P(S)$. If $\mathrm{lb}(S) \geq z_{\text{best}}$ backtrack.

(2) If $V_1(S) \neq \emptyset$, then tests are carried out to determine if S has a feasible completion. The basic test is as follows: Calculate

$$\sup_i = \sum_{k \in K(i)} \max_{j \in J_k \cap L(S)} \{a_{ij}\}, \quad \text{for } i \in V_1(S);$$

$\sup_i < b_i(S)$ backtrack.

(3) (a) If $\sup_i = b_i(S)$, then all variables whose coefficients determine $\sup_i$ can be set to 1 and all variables x_j, $j \in L(S)$ with $a_{ij} < 0$ can be set to 0; they can be fixed if $\sup_i$ is determined by a unique set of variables.

(b) If an element a_{ij} in $\sup_i$ is replaced by the next smaller admissible element of the special ordered set and the sum is less than $b_i(S)$, then x_j can be fixed to 1.

(c) If there exist elements $a_{ij} < 0$, $i \in V_1(S)$, $j \in L(S)$, such that $\sup_i + a_{ij} < b_i(S) \leq \sup_i$, then x_j can be fixed to 0.

(4) If no candidates where found in step 3 which can be set or fixed to 0 or 1, then a variable x_j, $j \in L(S)$ has to be selected to be set to 1. Several branching rules are possible. For most problems the following rule seemed to work best:

if $V_1(S) = \emptyset$, select x_j with $c_j = \min_{k \in L(S)} \{c_k\}$;

if $V_1(S) \neq \emptyset$, select the variable with the smallest cost coefficient from those variables which determine $\sup_i$.

(5) The candidates found in step 3 or the variable selected in step 4 are set or fixed to 0 or 1. This is called a forward step. The vectors s, st and x_S, the right hand side $b_i(S)$, $i \in M$, and the cost constant $z(S)$ are updated. Note that $L(S)$ is not updated explicitly but is stored implicitly via x_S, s and st. Further, if variable x_j from a special ordered set J_k is assigned a value 1 the other variables x_i, $i \in J_k$, $i \neq j$ are not set to 0 but remain free and only x_S is updated. This requires substantially less book keeping and storage space than explicit handling.

(6) If exactly one multiple choice constraint is still violated then the set of admissible candidates $L_k(S)$, $k \in V_2(S)$, is searched for a feasible completion. The search is sequential by increasing cost coefficients, thus the least cost completion is found first and the partial solution S is always fathomed.

(7) The enumeration process backtracks if a partial solution S can be fathomed. In this case the partial solution vector s is searched from right to left until an index j is found whose status is "set to β". The status of x_j is then replaced by "fixed to $1-\beta$" and x_S, $z(S)$, $b(S)$, s, and st are updated; all indices in s to the right of j change their status from "fixed" to "free". The enumeration stops when all elements of S have status "fixed".

For this implementation the efficiency of the basic enumeration was increased by using additional characteristics of the problem under consideration. For example, if constraints of the type $\sum_{j \in N} a_{ij} x_j \leq b_i$ with $a_{ij} \geq 0$ and $b_i \geq 0$ are present, the fact that these constraints must never be violated can be used advantageously in steps 2, 3, 4, and 6. Similarly, special tests were used for assembly-line-balancing problems. Finally, penalties can be calculated in steps 3 and 4 of the algorithm which reduces the number of branches significantly.

3. Data organization

The efficiency of any enumeration procedure depends critically on the organization and storage structure of the problem data. The coefficient matrix of realistic problems is, in general, large but sparse. It is, therefore, not possible to keep the entire matrix in core. In addition, storing all elements explicitly will require an excessive computational effort for the usual feasibility and branching tests.

Storing non-zero elements by rows, only, will reduce core requirements significantly and to some extend computation times for feasibility and branching tests. The updating of the right-hand-side, however, requires prohibitive search times.

For this implementation non-zero elements were stored and chained row-wise as well as column-wise. The list structure can be characterized as follows:

- variables are ordered by increasing cost coefficients within each special ordered set;
- the constraint matrix (1) is partitioned into positive elements and negative elements;
- the positive elements of the same row and special ordered set are chained in decreasing order of magnitude;
- the negative elements of each row are chained in increasing order of magnitude;
- for each column, the positive elements and the negative elements are chained;
- the multiple choice constraints are stored implicitly.

The storage of the coefficients of the constraint matrix (1) requires the following 5 arrays:

(a) value of element a_{ij};

(b) row index i;

(c) column index j;

(d) pointer to next smaller positive element in same row and special ordered set, or pointer to next larger negative element in same row;

(e) pointer to next non-zero element in same column.

The array (d) can be eliminated if elements are sorted in the appropriate order. In addition the following pointer arrays are used:

(f) largest positive element in row i and special ordered set k;

(g) smallest negative element in row i;

(h) first positive element in column j;

(i) first negative element in column j;

(k) first variable of special ordered set k.

Finally, the arrays s, M, σ_j, c_j and b have to be stored and one additional array is used which orders the variables by increasing cost coefficients within special ordered sets.

The list structure allows efficient testing as well as updating. To calculate sup, in step 2 of Fig. 1, for example, the vector s is searched sequentially for zero entries. Assume $s(k) = 0$; then pointer array (f) points directly to the largest positive element a_{ij} in row i and special ordered set k. If the status of variable x_j is free, then a_{ij} is an element of $\sup_i$; otherwise pointer array (d) is used to retrieve the next smaller element in row i and special ordered set k, etc.

Similarly, in step 5c of Fig. 1, row i is searched for negative coefficients a_{ij}. For this test, pointer array (g) points to the smallest negative element a_{ij}. If $\sup_i + a_{ij} < b_i(S)$ and $j \in L(S)$, then x_j is fixed to 0 and pointer array (d) is used to retrieve the second smallest element in row i, etc., until the test fails for the first time.

As a final illustration, in step 6 of Fig. 1, the admissible variables of the remaining violated multiple choice constraint are searched sequentially for a feasible completion. If $V_i(S) = \emptyset$ then for $j \in L(S)$ only $a_{ij} < 0$ have to be checked against $b_i(S) < 0$. In this case, pointer array (i) leads to the top of the chain of negative elements in column j. If $V_i(S) \neq \emptyset$, pointer array (h) leads to the chain of positive elements $a_{ij} > 0$ in column j which are checked against $b_i(S) > 0$.

4. Numerical results

A preliminary version of the algorithm was implemented on a CDC CYBER 72. Three types of test problems were generated and run for various problem sizes. Problem A is an assembly-line-balancing problem. A detailed description can be found in [18]. Problems of type B are distribution and warehouse allocation problems with side constraints. Problem B.1 is based on [9], however, additional side constraints were added to render the solution of [9] infeasible. The data for problems B.2–B.5 were generated randomly. Coefficients of the objective functions are uniformly distributed in the interval [1, 101]; coefficients of the general constraints are uniformly distributed in the interval [1, 51]. The right-hand-side coefficients of each problem were assigned values between 60 and 120. Problem C is a resource-constrained network scheduling problem. Table 1 summarizes the results.

Thangavelu and Shetty [20] developed an efficient algorithm for assembly-line-balancing problems without additional side constraints. They solved problem A in 4.8 sec. on the UNIVAC 1108. Solution times are difficult to compare; the UNIVAC 1108 is, in general, several times faster than the CYBER 72. Problem B.1 without side constraints was solved by DeMaio and Roveda [9] who designed a specialized algorithm to solve "pure" problems of this type. Their reported solution time was 1 sec. on the 1108. Finally, problem C was solved previously in 58 sec. on the IBM 370/158, using MPSX-MIP. For comparison, an attempt was made to solve all problems except B.3, B.4, and B.5 using CDC's LP-based system APEX III. This code has a feature to handle special ordered sets implicitly and efficiently. Problems B.1 and C were solved on the CYBER 72 in 19 seconds and 200 seconds CPU-time, respectively. All other problems could not be solved in 1 hour CPU-time; the feasible solutions which were found in 1 hour CPU-time did not contain the optimal solution in any of these cases.

Table 1.

n	number of 0-1 variables
m	number of constraints
p	number of multiple choice constraints
d	density of problem matrix
prob	number of problems solved
nodes	number of partial solutions investigated
solns	number of feasible solutions found
CPU	total CPU-time in sec., including input processing

Problem	n	m	p	d	prob	nodes		solns		CPU	
						min	max	min	max	min	max
A	450	117	45	4.1	1	—	1280	—	2		6.1
B.1	20	13	5	23.1	7	4	12	1	2	0.2	0.2
B.2	400	40	20	5.0	5	1237	7191	4	9	3.4	19.7
B.3	1000	65	25	3.1	1		141	—	2	—	1.6
B.4	1000	65	40	3.1	2	1075	6256	4	4	5.1	31.1
B.5	1000	70	50	2.9	1		22588		3		90.2
C	58	46	6	23.5	1	—	79	—	15*		1.5

* All optimal solutions enumerated

The test results are insufficient to draw any final conclusions. It appears, however, that even large problems of this special structure can be solved by implicit enumeration in reasonable CPU-time. The number of general constraints seem to have little influence on solution times as the increased computational effort is offset by tighter bounds. The number of special ordered sets, however, appears to be a limiting factor, as the computation times increase exponentially with the number of special ordered sets.

References

[1] S. Ashour and A. Char, Computational experience on zero-one programming approaches to various combinatorial problems, J.Op.Res.Soc. Japan 13 (1970) 78–108.

[2] K.R. Baker, Introduction to Sequencing and Scheduling, (Wiley, 1974).

[3] E. Balas, An additive algorithm for solving linear programs with zero-one variables, ORSA 13 (1965) 517–546.

[4] E. Balas, Discrete programming by the filter method, ORSA 15 (1967) 915–957.

[5] M. Balinski and K. Spielberg, Methods for integer programming, in: J. Aronofsky, ed., Progress in Operations Research Vol. III, (Wiley, New York, 1969) pp. 195–292.

[6] E.M.L. Beale and J.A. Tomlin, Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables, in: J. Lawrence, ed., Proc. 5th IFORS Conference (Wiley, New York, 1970).

[7] A.R. Brown, Selling television time: An optimization problem, Computer J. 12 (1969) 201–207.

[8] A.R. Brown, Optimum Packing and Depletion, (McDonald, London, 1971).

[9] A. De Maio and C. Roveda, An all zero-one algorithm for a certain class of transportation problems, ORSA 19 (1971) 1406–1418.

[10] B. Fleischmann, Computational experience with the algorithm of Balas, ORSA 15 (1967) 153–155.

[11] R.S. Garfinkel and G.L. Nemhauser, Integer Programming, (Wiley, New York, 1972).

[12] A.M. Geoffrion and G.W. Graves, Multicommodity distribution system design by Benders decomposition, Management Sci. 20 (1974) 822–844.
[13] A. Geoffrion, Integer programming by implicit enumeration and Balas' method, SIAM Review 9 (1967) 178–190.
[14] A. Geoffrion, An improved implicit enumeration approach for integer programming, ORSA 17 (1969) 437–454.
[15] F. Glover, A multiphase-dual algorithm for the zero-one programming problem, ORSA 13 (1965) 879–919.
[16] F. Glover and S. Zionts, A note on the additive algorithm of Balas, ORSA 13 (1965) 546–549.
[17] W. Healy, Jr., Multiple choice programming, ORSA 12 (1964) 122–138.
[18] M.D. Kilbridge and L. Wester, A review of analytical systems of line balancing, ORSA 10 (1962) 626–638.
[19] N.L. Lawrie, An integer linear programming model of a school time-tabling problem, Computer J. 12 (1969) 307–316.
[20] C. Lemke and K. Spielberg, Direct search algorithms for zero-one and mixed integer programming, ORSA 15 (1967) 892–914.
[21] C. Petersen, Computational experience with variants of the Balas algorithm applied to the selection of R. & D. projects, Management Sci. 13 (1967) 736–750.
[22] G.T. Ross and R.M. Soland, A branch and bound algorithm for the generalized assignment problem, Math. Programming 8 (1975) 91–103.
[23] H. Salkin, On the merit of generalized origin and restarts in implicit enumeration, ORSA 18 (1970) 549–555.
[24] H. Salkin, Integer Programming (Addison Wesley, Reading, MA, 1975).
[25] U. Suhl, Entwicklung von Algorithmen für ganzzahlige Optimierungsmodelle, unpublished dissertation, Freie Universität Berlin (1975).
[26] S.R. Thangavelu and C.M. Shetty, Assembly line balancing by 0–1 integer programming, AIIE Trans. III (1971) 64–70.
[27] N. Tuan, A flexible tree-search method for integer programming problems, ORSA 19 (1971) 115–119.

Annals of Discrete Mathematics 1 (1977) 403–414

ON SOME NONLINEAR KNAPSACK PROBLEMS

I. MICHAELI and M.A. POLLATSCHEK
Faculty of Industrial and Management Engineering, Technion, Haifa, Israel

Minimization of separable strictly convex function is considered with nonnegative integer variables when the sum of variables is constrained. Theorems concerning the condition for the optimum and properties of the optimal solution are presented. For a few types of functions this problem displays "periodic" properties similar to those in linear integer programming: The difference between the noninteger and integer solution is a function depending solely on the position of the noninteger solution inside a hypercube formed by the neighbouring integer points. Utilization of this property shortens drastically the search for the integer solution. In many cases the problem reduces to nonlinear 0/1 problem.

1. Introduction

Nonlinear integer programs have attracted less attention than their linear or nonlinear 0/1 counterparts. (See [3, 5] and the works referenced there for these two cases.) We are aware of references [1, 2, 6, 7, 8 and 10] only. If general theorems are desired, even the convex case appears to be quite intractable when there are more than one variable as has been pointed out recently [9].

Our aim is to eventually deal with the program wherein the minimand is separable and strictly convex and the constraints are linear. This paper is the first step toward this end: the constraint treated here is that the sum of variables is b. Thus, our problem is (P):

$$\text{minimize} \quad \sum_{i=1}^{n} f_i(x_i) = F(X), \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = b, \tag{2}$$

$$x_i \geq 0, \quad i = 1, 2, \ldots, n, \tag{3}$$

$$x_i \text{ is integer}, \quad i = 1, 2, \ldots, n. \tag{4}$$

where b is a positive integer and $f_i(x_i)$ is finite strictly convex for each i, for all the values x_i satisfying (2) and (3).

The authors have been motivated by a problem where $f_i(x_i) = c_i p_i^{x_i}$, which arises for example in allocating b (identical) weapons to n targets. Let c_i be the utility of destroying target i, $1 - p_i$ the probability of destruction by a single weapon, assuming independence among the weapons and additivity of utility one arrives (after trivial modifications) to (P) with the above $f_i(\,.\,)$ where x_i is the number of weapons allocated to target i.

It is hoped that (P) will serve as a vehicle to analyse the case where the constraints are linear but otherwise arbitrary.

In Section 2 an easily applicable necessary and sufficient condition is derived for the (integer) optimum of (P) (Theorem 1 and its corollary). Denote an (integer) optimal point of (P) by x^0 and the optimal point when (4) is disregarded by x^*. It is shown that either $x_i^0 \geq [x_i^*]$ for each i or $x_i^0 \leq [x_i^*]+1$ for each i, or both, when $[a]$ is the largest integer not exceeding a. It is easy to check whether both inequalities hold in which case (P) reduces to a 0/1 program which is less difficult to solve.

It is hoped that Theorems 1 and 2 can be extended to a more general program, although their proofs exploit heavily the properties of constraint (2).

For a few types of functions it can be shown that $x^0 - x^*$ is not a function of b. This is very similar to the phenomenon in asymptotic integer linear programs [4] and has not been previously observed in the literature for the nonlinear case. Thus, a general integer solution may be provided for an infinite number of right-hand sides.

2. Theorems

Theorem 1. *$x = (x_1, x_2, \ldots, x_n)$ is a solution of* (P) *if and only if it satisfies* (2), (3), (4) *and* (5):

$$f_i(x_i) + f_j(x_j) \leq f_i(x_i + m) + f_j(x_j - m) \tag{5}$$

for each pair i, j $(i = 1, \ldots, n;\ j = 1, \ldots, n;\ i \neq j)$ and each integer m such that

$$-x_i \leq m \leq x_j .$$

Proof. The necessity of (2), (3), (4) and (5) for optimum is trivial. Their sufficiency will be established by contradiction. Assume that x^0 is an optimal solution and x^1 is not: $F(x^1) > F(x^0)$, while both are feasible, i.e., satisfy (2), (3) and (4). Suppose that x^1 also satisfies (5) (x^0 clearly does). Denote one of them by x^a and the other by x^b as follows: Define

$$\alpha_i \doteq x_i^a - x_i^b \tag{6}$$

and order the variables and points so that

$$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n \tag{7}$$

$$\alpha_1 + \alpha_n \geq 0. \tag{8}$$

Note that this can be done without loss of generality and since both x^a and x^b satisfy (2) we have

$$\sum_{i=1}^{n} \alpha_i = 0. \tag{9}$$

Moreover, $x^a \neq x^b$ and (7) imply that

$$\alpha_1 > 0; \qquad \alpha_n < 0. \tag{10}$$

Two cases will be dealt with separately: the case where $\alpha_1 + \alpha_n = 0$ and the case where $\alpha_1 + \alpha_n < 0$. (By (8), these are the only possibilities).

Case 1. Assume $\alpha_1 + \alpha_n = 0$. Consequently,

$$x_1^a + x_n^a = x_1^b + x_n^b$$

and by assumption both x^a and x^b satisfy (5), hence

$$f_1(x_1^a) + f_n(x_n^a) \le f_1(x_1^b) + f_n(x_n^b), \tag{11}$$

$$f_1(x_1^b) + f_n(x_n^b) \le f_1(x_1^a) + f_n(x_n^a). \tag{12}$$

From (11) and (12) follows that

$$f_1(x_1^a) + f_n(x_n^a) = f_1(x_1^b) + f_n(x_n^b) \tag{13}$$

If $\alpha_1 = 1$ (and $\alpha_n = -1$) then, by (7) and (9), either $\alpha_i = 1$ or $\alpha_i = -1$ or $\alpha_i = 0$ for each i. From (7) and (9) it also follows that

$$\alpha_i + \alpha_{n+1-i} = 0$$

for each i, and hence analogously to (13):

$$f_i(x_i^a) + f_{n+1-i}(x_{n+1-i}^a) = f_i(x_i^b) + f_{n+1-i}(x_{n+1-i}^b).$$

Since for odd number of variables there must exist

$$\alpha_{(n+1)/2} = 0,$$

we have (for even or odd number of variables)

$$F(x^a) = F(x^b),$$

which is a contradiction to the initial assumption that $F(x^a) < F(x^b)$.

If $\alpha_1 > 1$ then by strict convexity of $f_1(x_1)$ and $f_n(x_n)$ and by (5) we have

$$f_1(x_1^a) + f_n(x_n^a) < f_1(x_1^a - \alpha_1) + f_n(x_n^a - \alpha_n)$$

or

$$f_1(x_1^a) + f_n(x_n^a) < f_1(x_1^b) + f_n(x_n^b),$$

which contradicts (13).

Case 2. Assume $\alpha_1 + \alpha_n < 0$. By assumption, both x^a and x^b satisfy (5):

$$f_1(x_1^a) + f_n(x_n^a) \le f_1(x_1^a - y) + f_n(x_n^a + y) \tag{14}$$

$$f_1(x_1^b) + f_n(x_n^b) \le f_1(x_1^b + z) + f_n(x_n^b - z) \tag{15}$$

for y and z integers, satisfying

$$x_n^a \leq y \leq x_1^a$$

$$-x^a \leq z \leq x_n^a.$$

Substituting (6) into (14) yields

$$f_1(x_1^b + \alpha_1) + f_n(x_n^a + \alpha_n) \leq f_1(x_1^b - \alpha_1 - y) + f_n(x_n^a + \alpha_n + y). \tag{16}$$

Eqs. (16) and (15) can be rearranged as follows:

$$f_n(x_n^a + \alpha_n) - f_n(x_n^b + \alpha_n - y) \leq f_1(x_1^a + \alpha_1 - y) - f_1(x_1^b - \alpha_1) \tag{17}$$

$$f_1(x_1^b) - f_1(x_1^a + z) \leq f_n(x_n^b - z) - f_n(x_n^b). \tag{18}$$

Let

$$y = z = \alpha_1. \tag{19}$$

Note that the substitution of (19) into (17) and (18) does not violate (3): By (6) and (10), α_1 is a positive integer, and

$$x^b + \alpha_1 - \alpha_1 - x_1^b \geq 0.$$

We still have to show that $x_1^b - \alpha_1$ is nonnegative. We have by (6)

$$x_n^b = x_n^a - \alpha_n = x_n^a + \alpha_1 \quad (\alpha_1 = -\alpha_n),$$

and since $\alpha_1 + \alpha_n \leq 0$,

$$x_n^b \geq x_n^a - \alpha_1,$$

$$x_n^b - \alpha_1 \geq x_n^a \geq 0,$$

which is the desired result.

Substitution of (19) into (17) and (18) results in equality between the right hand-side of (17) and the left hand-side of (18), implying:

$$f_n(x_n^b + \alpha_n) - f_n(x_n^a - \alpha_1 + \alpha_n) = f_n(x_n^b - \alpha_1) - f_n(x_n^a),$$

which can be rearranged as:

$$f_n(x_n^b + \alpha_n) + f_n(x_n^a) = f_n(x_n^b - \alpha_1) + f_n(x_n^a + \alpha_1 - \alpha_n). \tag{20}$$

By (7), (20) and the assumption $\alpha_1 + \alpha_n \leq 0$,

$$x_n^a + \alpha_n < x_n^b - \alpha_1 < x_n^b$$

$$x_n^b + \alpha_n < x_n^a + (\alpha_1 + \alpha_n) < x_n^a$$

and due to strict convexity of $f_n(x_n)$ we have

$$f_n(x_n^b - \alpha_1) < \lambda f_n(x_n^a) + (1 - \lambda) f_n(x_n^b + \alpha_n) \tag{21}$$

$$f_n(x_n^a + \alpha_1 + \alpha_n) < (1 - \lambda) f_n(x_n^a) + \lambda f_n(x_n^b + \alpha_n) \tag{22}$$

for $\lambda = 1 + \alpha_1/\alpha_n$ (note, that $0 < \lambda < 1$).

Summation of (21) and (22):

$$f_k(x_k^* - \alpha_1) + f_n(x_n^* + \alpha_1 - \alpha_n) < f_k(x_k^*) + f_n(x_n^* + \alpha_n)$$

contradicts (20).

These exhaust all the cases, and the proof of Theorem 1 is now complete. □

Corollary. x is a solution of (P) if and only if it satisfies (2) through (4) and

$$f_i(x_i) + f_j(x_j) \le f_i(x_i + m) + f_j(x_j - m) \tag{5}$$

for each pair i, j ($i = 1, \ldots, n$; $j = 1, \ldots, n$; $i \neq j$) and integer m:

$$\max\{-1, -x_i\} \le m \le \min\{x_j, 1\}. \tag{23}$$

Proof. The corollary differs from the theorem only in (23), which replaces the condition of the theorem

$$x_i \le m \le x_j.$$

However, this is possible by strict convexity of right hand-side of (5) as a function of m, which follows from strict convexity of f_i and f_j. □

Consider the problem

$$\text{minimize}\,\{f_i(y_i) + f_j(y_j) \mid y_i - y_j = \beta;\ y_i, y_j \in R\}.$$

Let $y^* = (y_i^*, y_j^*)$ be the solution; clearly y^* is a function of β.

Lemma 1. *y_i^* and y_j^* are monotone nondecreasing functions of β, while*

$$\beta_1 < \beta_2 \Longrightarrow \begin{cases} y_i^*(\beta_2) - y_i^*(\beta_1) \le \beta_2 - \beta_1, \\ y_j^*(\beta_2) - y_j^*(\beta_1) \le \beta_2 - \beta_1. \end{cases} \tag{24}$$

If, moreover, f_i and f_j are differentiable, then y_i^* and y_j^* are monotone increasing in β, while

$$\beta_1 < \beta_2 \Longrightarrow \begin{cases} y_i^*(\beta_2) - y_i^*(\beta_1) < \beta_2 - \beta_1, \\ y_j^*(\beta_2) - y_j^*(\beta_1) < \beta_2 - \beta_1. \end{cases} \tag{24'}$$

Proof. Consider β_1 and β_2, $\beta_1 < \beta_2$ and the corresponding optimal $y_1^* = (y_{i1}^*, y_{j1}^*)$ and $y_2^* = (y_{i2}^*, y_{j2}^*)$. Now

$$\beta_1 < \beta_2 \Longrightarrow y_{i1}^* + y_{j2}^* < y_{i2}^* + y_{j1}^*. \tag{25}$$

Lagrangian optimality conditions require that

$$f_i'(y_{i1}^*) = f_j'(y_{j1}^*); \quad f_i'(y_{i2}^*) = f_j'(y_{j2}^*). \tag{26}$$

where f'_i denote a point from the proper subdifferentials at points y^*_{i1} and y^*_{i2} resp. Suppose that $y^*_{i1} > y^*_{i2}$. Consequently:

$$y^*_{i1} > y^*_{i2} \Longrightarrow f'_i(y^*_{i1}) > f'_i(y^*_{i2})$$

$$\Longleftrightarrow f'_j(y^*_{j1}) > f'_j(y^*_{j2})$$

$$\Longrightarrow y^*_{j1} > y^*_{j2}.$$

The implications are due to the fact that f is strictly convex, while the equivalence follows from (26). Hence,

$$y^*_{i1} + y^*_{j1} > y^*_{i2} + y^*_{j2},$$

which contradicts (25) and proves that y^*_i are monotonic nondecreasing functions of β for each i.

If the differentials exist (and denoted by f'_i and f'_j) then supposing $y^*_{i1} \geq y^*_{i2}$ implies:

$$y^*_{i1} \geq y^*_{i2} \Longleftrightarrow f'_i(y^*_{i1}) \geq f'_i(y^*_{i2})$$

$$\Longleftrightarrow f'_j(y^*_{j1}) \geq f'_j(y^*_{j2})$$

$$\Longleftrightarrow y^*_{j1} \geq y^*_{j2}.$$

analogously to the previous case, which similarly contradicts (25) proving the monotone increasing property.

Eqs. (24) and (24') follows from the monotone property and that the sum of y^*_i and y^*_j must be β. ∎

Theorem 2. *Let $x^* = (x^*_1, \ldots, x^*_n)$ be the (continuous) solution of (1), (2) and (3), and let $x^0 = (x^0_1, \ldots, x^0_n)$ be the (integer) solution of (P). Then either for each i*

$$x^0_i < x^*_i + 1$$

or for each i

$$x^0_i > x^*_i - 1$$

or both.

Proof. By Theorem 1, the optimality of x^0 may be established by looking at pairs of variables. Let us look at the variables x_i and x_j. Define

$$\beta^* \doteq x^*_i + x^*_j$$

$$\beta^0 \doteq x^0_i + x^0_j$$

$$\delta \doteq \beta^0 - \beta^*.$$

Let $y^* = (y_i^*, y_j^*)$ be the optimal solution of

$$\text{minimize } \{f_i(y_i) + f_j(y_j)\}$$
$$\text{subject to } y_i + y_j = \beta^*.$$

If y^* is integer we must have $x_i = y_i^*$, since x^* is optimal, therefore the following is implied:

$$y_i^* - 1 < x_i^* < y_i^* + 1. \tag{27}$$

If y^* is noninteger note that $[y_i^*] - k$ and $[y_j^*] + 1 - k$ consist of a feasible solution to the above minimization for any integer k. By optimality of y^* and strict convexity of $f_i(.)$ and $f_j(.)$:

$$f_i(y_i^*) + f_j(y_j^*) < f_i([y_i^*]) + f_j([y_j^*] + 1) < f_i([y_i^*] - m) + f_j([y_j^*] - 1 + m),$$
$$f_i(y_i^*) - f_j(y_j^*) < f_i([y_i^*] + 1) + f_j([y_j^*]) < f_i([y_i^*] + 1 + m) - f_j([y_j^*] - m),$$

for any $m > 0$, integer. Now, by Theorem 1,

$$f_i(x_i^*) + f_j(x_j^*) \leq f_i(x_i + n) - f_j(x_j^* - n)$$

for any integer n.

By comparing the last three inequalities it is apparent that either $x_i^* = [y_i^*]$ or $x_i^* = [y_i^*] + 1$. This again implies (27).

Let us now deal with three cases: The case where $\delta > 0$, the case where $\delta < 0$ and the case where $\delta = 0$.

Case 1. For $\delta > 0$ we have by Lemma 1

$$x_i' < y_i^* < x_i^* - \delta. \tag{28}$$

By combining (27) with (28) we obtain

$$x_i' - 1 < x_i^* < x_i' - \delta - 1, \tag{29}$$

and by the same way

$$x_j' - 1 < x_j^* < x_j' + \delta + 1. \tag{30}$$

Case 2. For $\delta < 0$ we have by Lemma 1

$$x_i' - \delta \leq y_i^* \leq x_i^*. \tag{31}$$

By combining (27) with (31) we obtain

$$x_i' - (\delta - 1) < x_i^* < x_i' - 1, \tag{32}$$

and by the same way

$$x_j^* - (\delta + 1) < x_j' < x_j^* - 1. \tag{33}$$

Case 3. For $\delta = 0$ we have $x_i^* = y_i^*$, so by (27) we have

$$x_i^* - 1 < x_i' < x_i^* - 1. \tag{34}$$

and also

$$x_j^* - 1 < x_j^0 < x_j^* + 1. \tag{35}$$

Let us now define the correction t_j of the variable x_j by

$$t_j = x_j^0 - x_j^*,$$

and let us arrange the variables in decreasing order of the t_j's such that

$$t_1 \geq t_2 \geq \cdots \geq t_n.$$

Since $\sum_{j=1}^{n} t_j = 0$, we have $t_1 \geq 0$ and $t_n \leq 0$. Note that the value of δ for the variables x_1 and x_n is

$$\delta = t_1 + t_n.$$

In the case where $t_1 + t_n \geq 0$ we have

$$t_1 + t_j \geq 0, \quad j = 2, \ldots, n,$$

and by (29), (30), (34) and (35) we get

$$x_j^0 > x_j^* - 1$$

for each j.

In the case where $t_1 + t_n < 0$ we have

$$t_j + t_n < 0, \quad j = 1, \ldots, n - 1,$$

and by (32), (33) we get

$$x_j^0 < x_j^* + 1$$

for each j. This completes the proof. □

Corollary. If there exists at least one variable for which $x_j^0 > x_j^* - 1$, there exist at least two variables for which $x_j^0 = [x_j^*]$, and if there exists at least one variable for which $x_j^0 < x_j^* + 1$, there exist at least two variables for which $x_j^0 = [x_j^*] + 1$. ($[x_j^*]$ is defined as the greatest integer which is not greater than x_j^*).

Proof. Define

$$e_j \triangleq x_j^* - [x_j^*].$$

By (2), $\sum_{j=1}^{n} x_j^* = \sum_{j=1}^{n} [x_j^*] + \sum_{j=1}^{n} e_j = b$. Since $[x_j^*]$ and b are integers, $\sum_{j=1}^{n} e_j$ must also be an integer. Since $e_j < 1$ for each j,

$$\sum_{j=1}^{n} e_j \leq n - 1. \tag{36}$$

Assume $x_1^0 > x_1^* - 1$, say

$$x_1^0 = [x_1^*] + t_1, \quad t_1 \geq 0. \tag{37}$$

By (2), $\sum_{i=1}^{n} x_i^\circ = b$ and we have

$$\sum_{i=1}^{n} x_i^\circ = \sum_{i=1}^{n} [x_i^*] + \sum_{i=1}^{n} \varepsilon_i. \tag{38}$$

Substitution of (37) into (38) yields

$$\sum_{i\ne k} x_i^\circ + l = \sum_{i\ne k} [x_i^*] + \sum_{i=1}^{n} \varepsilon_i.$$

By (36),

$$\sum_{i\ne k} x_i^\circ + l \le \sum_{i\ne k} [x_i^*] + (n-1)$$

or

$$\sum_{i\ne k} x_i^\circ \le \sum_{i\ne k} [x_i^*] + (n-1-l).$$

By (37), $n-1-l \le n-3$, hence

$$\sum_{i\ne k} x_i^\circ \le \sum_{i\ne k} [x_i^*] + (n-3). \tag{39}$$

By Theorem 2 and by (37), for each i, $x_i^\circ \ge [x_i^*]$, and by (39) at most $n-3$ variables may be at their optimal value with values greater than $[x_i^*]$. This leaves at least two variables for which $x_i^\circ = [x_i^*]$.

The second part of the corollary will be established in a similar way. Assume

$$x_k^\circ = ([x_k^*] + 1) - l\,; \quad l \ge 2. \tag{40}$$

By (2),

$$\sum_{i=1}^{n} x_i^\circ = \sum_{i=1}^{n} [x_i^*] + \sum_{i=1}^{n} \varepsilon_i$$

$$\sum_{i=1}^{n} x_i^\circ = \sum_{i=1}^{n} ([x_i^*] + 1) - \sum_{i=1}^{n} (1-\varepsilon_i). \tag{41}$$

Substitution of (40) into (41) yields

$$\sum_{i\ne k} x_i^\circ - l = \sum_{i\ne k} ([x_i^*] + 1) - \sum_{i=1}^{n} (1-\varepsilon_i).$$

Since $\sum_{i=1}^{n} (1-\varepsilon_i) \le n-1$,

$$\sum_{i\ne k} x_i^\circ - l \ge \sum_{i\ne k} ([x_i^*] + 1) - (n-1)$$

or

$$\sum_{i\ne k} x_i^\circ \ge \sum_{i\ne k} ([x_i^*] + 1) - (n-1-l).$$

By (40), $n - 1 - l \leq n - 3$, hence,

$$\sum_{i \in S} x_i^* \geq \sum_{i \in S} ([x_i^*] - 1) - (n - 3). \tag{42}$$

By Theorem 2 and by (40), for each i, $x_i^* \leq [x_i^*] + 1$, and by (42) at most $n - 3$ variables may be in their optimal values with values lower than $[x_i^*] + 1$. This leaves at least two variables for which $x_i^* = [x_i^*] + 1$. $\square$

Lemma 2. *If $f_1(y_1)$ and $f_2(y_2)$ are differentiable and strictly convex, then $G(\beta)$, defined as*

$$G(\beta) = \min_{y_1, y_2} \{ f_1(y_1) + f_2(y_2) \mid y_1 + y_2 = \beta;\ y_1, y_2 \in R \},$$

is also differentiable and strictly convex in β.

Proof. Is straightforward, therefore omitted. $\square$

The recursive application of Lemma 2 to Lemma 1 for differentiable f_i, $i = 1, 2, \ldots, n$, implies the following:

Corollary. *Denote by x^* the optimal (continuous) solution of (1)–(2). Then x^* is a monotone nondecreasing function of b.*

Assume the existence of $\bar{b}$ so that $x^* \geq 0$ (the inequality is taken componentwise). Then, by the corollary, any right-hand side, $b \geq \bar{b}$ implies $x^* \geq 0$, and (3) is automatically satisfied.

This is analogous to the asymptotic integer-linear programs [4] where the nonnegativity requirement is also assumed to hold. The analogy — at least for a few nonlinear functions, $f_i(x_i)$, — is deeper: the difference between noninteger and integer solutions is independent of the right-hand side. This will be shown for $f_i(x_i) = c_i p_i^{x_i}$:

By Theorem 1:

$$f_i(x_i^0) - f_i(x_i^0 + m) = c_i p_i^{x_i^0} - c_i p_i^{x_i^0 + m} \leq c_j p_j^{x_j^0 - m} - c_j p_j^{x_j^0}$$
$$= f_j(x_j^0 - m) - f_j(x_j^0)$$

If $b \geq \bar{b}$, (3) may be discarded from the program comprising (1), (2) and (3) and at the continuous) optimum, x^*:

$$f_i'(x_i^*) = c_i p_i^{x_i^*} \ln p_i = c_j p_j^{x_j^*} \ln p_j = f_j'(x_j^*)$$

Dividing the above inequality's sides by the corresponding sides of this equality and simplifying we obtain

$$p_i^{z_i - m}\left(\frac{p_i^m - 1}{\ln p_i}\right) \le p_j^{z_j}\left(\frac{p_j^m - 1}{\ln p_j}\right), \tag{43}$$

which, by the notation: $z_i \triangleq x_i^0 - [x_i^0]$, $\varepsilon_i \triangleq x_i^* - [x_i^*]$, may be written as:

$$p_i^{\varepsilon_i}\left(\frac{p_i^m - 1}{p_i^{z_i}\ln p_i}\right) \le p_j^{\varepsilon_j}\left(\frac{p_j^m - 1}{p_j^{m+z_j}\ln p_j}\right).$$

Note that a necessary and sufficient condition for the optimum is that this inequality holds for each pair i, j and integer m. z is a function of the problem's parameters and ε: the position of x^* in the hypercube $\{x \mid [x_i^*] \le x_i \le [x_i^*] + 1\}$. Two different right-hand sides b', b'' yielding the same ε induce also the same z or $x^0 - [x^*]$, or, equivalently, the same $x^0 - x^*$ (as ε is equal for b' and b'', by assumption). There are a few other functions for which z is determined by ε only.

This observation depends on the fact that (43) is a function of $x^0 - x^*$, m and the parameters of the problem only. It can be generalized as follows:

Theorem 3. *Let $f_i(x_i)$ be differentiable, denote its differential by $f_i'(x_i)$. Assume that there exists a $\bar{b}$ such that for $b \ge \bar{b}$ (3) is satisfied by the solution of (1)–(2). Assume the existence of a function $H(\ldots,\ldots)$ and of n functions $\phi_i(\ldots)$ such that $H(\ldots,\ldots)$ is monotone increasing in its first argument and*

$$H(f_i(x_i) - f_i(x_i - m), f_i'(y_i), m) = \phi_i(m, x_i - y_i).$$

Then $x_i^0 - x_i^*$ depends only on $x_i^* - [x_i^*]$ for each i (and not on b) when $b \ge \bar{b}$.

Proof. If $b \ge \bar{b}$, (3) may be disregarded from the program comprising (1), (2) and (3) and at the (continuous) optimum, x^*:

$$f_i'(x_i^*) = f_j'(x_j^*).$$

Theorem 1 which can be written as

$$f_i(x_i^0) - f_i(x_i^0 + m) \le f_j(x_j^0 - m) - f_j(x_j^0),$$

is equivalent with

$$\phi_i(m, x_i^0 - x_i^*) \le \phi_j(m, x_j^0 - x_j^* - m)$$

by the required property of H and the equality of the differentials. The last inequality, which is necessary and sufficient to the (integer) optimum, implies that $x^0 - x^*$ depends only on $x^* - [x^*]$ (when $[\,.\,]$ is applied componentwise). $\square$

Finally we illustrate Theorem 3 for the function $f_i(x_i) = u_i x_i^2 - v_i x_i + w_i$:

$$H(f_i(x_i^0) - f_i(x_i^0 - m), f_i'(x_i^*), m) =$$

$$= f_i(x_i^0) - f_i(x_i^0 - m) - m f_i'(x_i^*)$$

$$= u_i(x_i^0)^2 - v_i(x_i^0) + w_i - [u_i(x_i^0 - m)^2 + v_i(x_i^0 - m) + w_i] - m[2u_i x_i^* + v_i]$$

$$= 2mu_i(x_i^0 - x_i^*) - u_i m^2 = \phi_i(x_i^0 - x_i^*, m).$$

Acknowledgements

The authors wish to express their gratitude to the anonymous referees who helped to make this communication clearer, correct and readable.

References

[1] E. Balas, Duality in discrete programming II. The quadratic case, *Management Sci.* 16 (1969) 14–32.
[2] R.E. Burkard, A method for mixed-integer convex programming, in: *Proc. Fourth Conf. on Probability Theory* (Editura Academiei Republicii Socialiste Romania, 1973).
[3] A.M. Geoffrion and R.E. Marsten, Integer-programming algorithms, *Management Sci.*, 18 (1972) 465–491.
[4] R.E. Gomory, On the relation between integer and non-integer solutions to linear programs, *Proc. Nat. Sci., USA* 53 (1965) 260–265.
[5] P.L. Hammer, Boolean procedures for bivalent programming, in: P.L. Hammer and G. Zoutendijk, eds., *Mathematical Programming in Theory and Applications* (North-Holland, Amsterdam, 1974).
[6] B. Korte, W. Krelle and W. Oberhofer, Ein Lexikographischer Suchalgorithmus zur Lösung allgemeiner ganzzahliger Programmierungsaufgaben, *Unternehmensforschung* 13 (1969) 73–98, 171–192.
[7] H.P. Kunzi and W. Oettli, Integer quadratic programming, in: R.L. Graves and P. Wolfe, eds., *Recent Advances in Mathematical Programming* (McGraw-Hill, New York, 1963).
[8] B.L. Miller, On minimizing non-separable functions defined on integers with an inventory application, *SIAM J. Appl. Math.*, 21 (1971) 166–185.
[9] A. Washburn, A note on integer maximization of unimodal functions, *Operations Res.* 23 (1975) 353–360.
[10] C. Witzgall, An all-integer programming algorithm with parabolic constraints, *J. SIAM*, 11 (1963) 855–871.

Annals of Discrete Mathematics 1 (1977) 415-419
© North-Holland Publishing Company

THE MINIMAL INTEGRAL SEPARATOR OF A THRESHOLD GRAPH

James ORLIN

Department of Operations Research, Stanford University, Stanford, California 94305, U.S.A.

A graph is called threshold if there exists a real number b and real numbers a_i associated with its vertices w_i such that $\sum_{i \in S} a_i \le b$ holds iff S is a stable (independent) set of vertices. The vector $(a_1, \ldots, a_n, b)$ associated to a threshold graph is called an integral separator if $a_i + a_j \ge b + 1$ for every edge (w_i, w_j). A simple algorithm is presented to determine for a given threshold graph its (unique) integral separator which minimizes b.

Let G be a loopless finite graph without multiple edges. If w is a vertex of G, let $d(w)$ be the degree of w. The edge joining vertices u and w will be denoted as (u, w).

Graph G is said to *have property P* if for every two vertices u, v such that (u, v) is an edge, and for every pair of vertices u^*, v^* with $d(u^*) \ge d(u)$ and $d(v^*) \ge d(v)$, (u^*, v^*) is an edge. In this definition it is possible that $u^* = u$ or that $v^* = v$.

It has been shown in [1] that graph G has property P iff it is a threshold graph.

Suppose G is a threshold graph with vertices $w_1, w_2, w_3, \ldots, w_n$. For $I \subseteq \{1, 2, \ldots, n\}$ let $S_I = \{w_i \mid i \in I\}$. Let $A = (a_1, a_2, \ldots, a_n)$ be a real vector and let b be a real number. The pair $[A : b]$ is said to *separate G integrally* if the following holds:

(1) $a_i \ge 0$ for $i = 1, \ldots, n$;

(2) $\sum_{i \in I} a_i \le b$ iff S_I is a stable (independent) set of vertices;

(3) $\sum_{i \in I} a_i \ge b - 1$ iff S_I is a non-stable set of vertices.

It was shown in [1] that a graph G is threshold iff there exists a pair $[A : b]$ which separates G integrally.

The following algorithm determines for a threshold graph G a hyperplane $[A^*, b^*]$ which separates G integrally and such that b^* is minimum. It will also be shown that it is the unique hyperplane with minimum b.

Algorithm A.

Step 0: Relabel the vertices as $w_1, \ldots, w_n$ such that $d(w_1) \le d(w_2) \le \cdots \le d(w_n)$.

Step 1: Let t = minimum index such that (w_t, w_{t+1}) is an edge of G. [If no such t exists let $a_i^* = 0$ for $i = 1$ to n and let $b^* = 0$. Then exit from algorithm.]

Step 2: If $d(w_1) = 0$ let $a_1^* = 0$. If $d(w_1) \ge 1$ let $a_1^* = 1$.

Step 3: For $i = 2$ to t: if $d(w_i) = d(w_{i-1})$ then let $a^*_i = a^*_{i-1}$; if $d(w_i) > d(w_{i-1})$ then let $a^*_i = 1 + a^*_1 + a^*_2 + \cdots + a^*_{i-1}$.

Step 4: Let $b^* = a^*_1 + a^*_2 + \cdots + a^*_t$.

Step 5: For $i = t + 1$ to n let s_i be the minimum index such that (w_i, w_{s_i}) is an edge. Then let $a^*_i = b^* - a^*_{s_i} + 1$.

Example. Let G be the graph in Fig. 1. Table 1 shows how the algorithm worked.

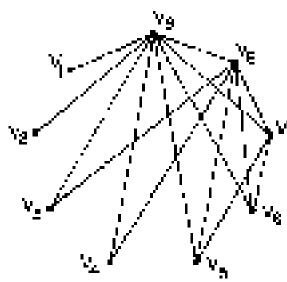


Fig. 1

Table 1

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
$d(V_i)$	1	1	2	2	3	3	4	6	8
a^*_i	1	1	3	3	9	9	18	24	26
defining step of algorithm	2	3	3	3	3	3	5	5	5

$t = 6$ as defined in step 1.
$b^* = 26$ as defined in step 4.

Proposition 1. *$[A^*; b^*]$ as constructed in algorithm A does separate the threshold graph G integrally.*

Proof. Assume that the vertices have already been relabeled such that $d(w_1) \leq d(w_2) \leq \cdots \leq d(w_n)$.

Case 0: Algorithm A exited at step 1 after labeling $a^*_i = 0$ for $i = 1$ to n.

Claim. G has no edges. Else consider edge (w_i, w_j) of G such that $i < j$. Then $i \leq n - 1$ and $j \leq n$. From this it follows that $d(w_{n-1}) \geq d(w_i)$ and $d(w_n) \geq d(w_j)$. But G has property P. Thus (w_{n-1}, w_n) is an edge. Thus in algorithm A $t \leq n - 1$. This contradicts that the algorithm exited at step 1. Hence G has no edges. It follows from the definitions that $[A^*; b^*]$ does separate G integrally in this case.

Now assume that algorithm A exited at step 5 with $[A^*; b^*]$ which does not separate G integrally.

Case 1: There exists a stable set S_t such that $\sum_{i \in T} a_i^* > b^*$. Let $T = \{j_1, j_2, \ldots, j_k\}$ with $j_1 < j_2 < \cdots < j_k$. If $j_k < t$ then $\sum_{i \in T} a_i^* \leq \sum_{i=1}^{t-1} a_i^* = b^*$ and we have a contradiction. Thus we may assume that $j_k \geq t - 1$. Let $q = s_{j_k}$ as chosen in step 5 of algorithm A. Thus (w_q, w_{j_k}) is an edge of G. If $q < j_{k-1}$ then $d(w_q) \leq d(w_{j_{k-1}})$. Since G has property P, this would mean that $(w_{j_{k-1}}, w_{j_k})$ is an edge of G, contradicting that S_t is stable. Hence we may assume that $q > j_{k-1}$. But now by construction of $[A^*, b^*]$ we have:

$$\sum_{i \in T} a_i^* \leq a_{j_k}^* + \sum_{i=1}^{q-1} a_i^* - a_{j_k}^* + (a_q^* - 1) = b^*.$$

Thus for all stable sets S_t the proposition is true.

Case 2. There exists a non stable set S_t such that $\sum_{i \in T} a_i^* < b^* + 1$.

Then S_t contains vertices w_i, w_j such that (w_i, w_j) is an edge. Assume that $i < j$. If $j < t$ then $i < t - 1$ and $d(w_i) \geq d(w_{t-1})$ and $d(w_j) \leq d(w_t)$. Since G has property P, this would imply that (w_{t-1}, w_t) is an edge, which is a contradiction. Hence we may assume that $j \geq t$. Then by the choice of S_j in step 5 it follows that $i \geq s_j$. But then

$$\sum_{i \in T} a_i^* \geq a_j^* - a_i^* \geq a_{s_j}^* - a_j^* = b^* + 1.$$

Thus the proposition is true.

Proposition 2. *Let $[A ; b]$ be any hyperplane that separates G integrally, where $A = (a_1, a_2, \ldots, a_t)$. Then for all i from 1 to t it is true that $a_i \geq a_i^*$.*

Proof. Once again assume $d(w_1) \geq d(w_2) \geq \cdots \geq d(w_t)$. If $d(w_1) = 0$ than $a_1^* = 0$ which is minimum by definition. Else there exists w_j such that (w_1, w_j) is an edge. Thus in any hyperplace $[A ; b]$ which separates G integrally we must have that $a_j \leq b$ and $a_1 + a_j \geq b + 1$. This implies that $a_1 \geq 1$. Thus $a_1 \geq a_1^* = 1$.

Assume inductively that a_i^* is minimum for $i = 1$ to $k - 1$ for $k \leq t$. It will be shown that a_k^* is also minimum.

Suppose $d(w_{k-1}) = d(w_k)$. Then since G has property P, w_{k-1} and w_k are adjacent to the same other vertices. By symmetry and by the induction hypothesis $a_k \geq a_{k-1}^*$. Since $a_{k-1}^* = a_k^*$ we have that $a_k \geq a_k^*$ and that a_k^* is thus minimum.

Suppose instead that $d(w_{k-1}) < d(w_k)$. Choose q to be the minimum index such that w_q is adjacent to w_k but not to w_{k-1}. Since G has property P, w_q is not adjacent to any w_i for $i = 1, \ldots, k - 1$; it is also true that no two vertices in $S - \{w_1, w_2, \ldots, w_{q-1}\}$ are adjacent. Thus in any hyperplane $[A ; b]$ we have

$$a_q + a_k \geq b + 1 \quad \text{and}$$

$$\sum_{i=1}^{q-1} a_i + a_q \leq b.$$

It follows that

$$a_k \ge 1 + \sum_{i=1}^{k-1} a_i \ge 1 + \sum_{i=1}^{k-1} a_i^* = a_k^*.$$

Corollary. *The value for b^* is also minimum.*

Proof. $\{w_1, w_2, \ldots, w_r\}$ is a stable set. Thus

$$b \ge \sum_{i=1}^{r} a_i \ge \sum_{i=1}^{r} a_i^* = b^*.$$

Proposition 3. *The algorithm constructs the unique $[A:b]$ which separates G integrally with minimum b.*

Proof. Suppose $[A:b^*]$ separates G integrally. Since

$$b^* \ge \sum_{i=1}^{r} a_i \ge \sum_{i=1}^{r} a_i^* = b^*$$

it follows that $a_i = a_i^*$ for $i = 1$ to r.

For $i = r+1, r+2, \ldots, n$ we have that

$$a_i + a_{s_i}^* \ge b^* + 1$$

$$a_i + \sum_{j=1}^{t_i} a_j^* \le b^*.$$

By construction

$$\sum_{j=1}^{t_i} a_j^* + 1 = a_{s_i}^*.$$

Thus $a_i = b^* - a_{s_i}^* + 1 = a_i^*$.

Proposition 4. *The hyperplane $[A^*; b^*]$ is also the solution to the following linear program:*

$$\min b$$

$$\text{s.t.} \sum_{i=1}^{r} a_i \le b \tag{1}$$

and, for $i = r+1$ to n,

$$a_i + a_{s_i} \ge b + 1 \tag{2}$$

$$a_i + \sum_{j=1}^{t_i} a_j \le b \tag{3}$$

where s_i and t_i are chosen as in the algorithm.

Proof. By (2) and property P for any non-stable set S_i, $\sum_{j \in S_i} a_j \geq b + 1$. If S_i is stable then either $I \subseteq \{1, 2, 3, \ldots, t\}$ or else $I \subseteq \{1, 2, \ldots, s_i - 1, j\}$ for some j. In either case by (1) and (3) we must have that $\sum_{j \in I} a_j \leq b$.

Acknowledgement. Partial support through NRC (Grant A 8552) is gratefully acknowledged.

Reference

[1] V. Chvátal and P.L. Hammer, Aggregation of Inequalities in Integer Programming, Ann. Discrete Math. 1 (1977) 145–162.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 421–434

ON THE COMPLEXITY OF SET PACKING POLYHEDRA*

Manfred W. PADBERG

Graduate School, Faculty of Business Administration, New York University, New York, NY 10006, U.S.A.

We review some of the more recent results concerning the facial structure of set packing polyhedra. Utilizing the concept of a facet producing graph we give a method that can be used repeatedly to construct (arbitrarily) complex facet-producing graphs. A second method, edge-division, is used to further enlarge the class of facet-defining subgraphs.

1. Introduction

We consider the set packing problem (SP)

$$\begin{aligned} &\max \ cx \\ &\qquad Ax \le e \qquad\qquad (\mathrm{SP}) \\ &\qquad x_j = 0 \quad \text{or} \quad 1 \text{ for } j = 1, \ldots, n \end{aligned}$$

where A is a $m \times n$ matrix of zeros and ones, $e^T = (1, \ldots, 1)$ is a vector of m ones and c is a vector of n (arbitrary) rational components. This class of combinatorial optimization problems has recently received much attention, both as a problem of considerable practical interest — see e.g. [3, 10, 22] for recent survey articles for this problem and its close relatives — as well as a combinatorial programming problem that captures most of the difficulties and computational complexities that are present in the general zero-one programming problem [2, 10, 11, 15].

In this paper we extend the class of "strongest" cutting planes or facets known for problem (SP) and show that for set packing problems of sufficiently large size arbitrarily "complex" valid inequalities can be constructed that, however, are facets of the convex hull of (integer) solutions to (SP), i.e. belong to the class of linear inequalities that *uniquely* define the convex hull of solutions to (SP). Without restriction of generality, we will assume throughout the paper that A does not have any zero column or zero row. Denote by $P = P(A, e)$ the polyhedron given by the feasible set of the linear programming problem associated with (SP), i.e.

$$P(A, e) = \{x \in \mathbf{R}^n \mid Ax \le e, x \ge 0\}. \qquad (1.1)$$

* Parts of this paper were presented in preliminary form at the NATO Advanced Study Institute Symposium on Combinatorial Programming held in Versailles, France, September 1974.

Furthermore, let $P_I = P_I(A, e)$ denote the set packing polyhedron, i.e. the convex hull of integer points of P

$$P_I(A, e) = \operatorname{conv}\{x \in P(A, e) \mid x \text{ integer}\} \tag{1.2}$$

By the theorem of Weyl [25], there exists a finite system of linear inequalities whose solution set coincides with P_I, i.e.,

$$P_I = \{x \in \mathbb{R}^n \mid Hx \leq h, x \geq 0\} \tag{1.3}$$

for some appropriate matrix H and vector h. Some research activity has recently focused on identifying part (or all) of those linear inequalities that define P_I, see [5, 16, 17, 18, 20, 21, 23, 24]. This interest is motivated in part by the desire to use linear programming duality in proving optimality — with respect to the linear form cx — of a given extreme point of P_I. As every extreme point of P_I is also an extreme point of P and hence of any polyhedron $\bar{P}$ satisfying $P_I \subset \bar{P} \subset P$, it is generally sufficient to work with a *partial* — rather than a complete — linear characterization of P_I. More precisely, one is interested in finding part (or all) of the linear inequalities that define facets of P_I. Note that $\dim P = \dim P_I = n$, i.e. both P and P_I are fully dimensional. As customary in the literature, we will call an inequality $\pi x \leq \pi_0$ a *facet* of P_I if (i) $\pi x \leq \pi_0$ for all $x \in P_I$ and (ii) there exist n affinely independent vertices x^i of P_I such that $\pi x^i = \pi_0$ for $i = 1, \ldots, n$. One readily verifies that each inequality $x_j \geq 0$, $j \in N$, is a (trivial) facet of P_I, where $N = \{1, \ldots, n\}$.

A construction that has proved useful in identifying facets of P_I is the intersection graph associated with the zero-one matrix A defining P. Denote by a_j the jth column of the $m \times n$ matrix A. The *intersection graph* $G = (N, E)$ of A has one node for every column of A, and one (undirected) edge for every pair of nonorthogonal columns of A, i.e. $(i, j) \in E$ iff $a_i a_j \geq 1$. One verifies readily that the weighted node packing problem (NP) on G for which the node weights equal c_j for $j = 1, \ldots, n$ is equivalent to (SP), i.e. (NP) has the same solution set and set of optimal solutions as the problem (SP). (The weighted *node packing problem* (NP) on a finite, undirected, loopless graph G is the problem of finding a subset of mutually non-adjacent vertices of G such that the total weight of the selected subset is maximal. See [6, 10, 15, 18] for more detail on the stated equivalence.) This observation is very useful as it permits one to restrict attention to node packing problems in certain subgraphs of G when one tries to identify facets of P_I.

2. Facet producing subgraphs of intersection graphs

Let $\pi x \leq \pi_0$ be a non-trivial facet of P_I. We can assume without loss of generality that both π_j, $j \in N$, and π_0 are integers. From the non-negativity of A, it follows readily that $\pi_j \geq 0$ for all $j \in N$ and $\pi_0 > 0$. For suppose that $N^- = \{j \in N \mid \pi_j < 0\} \neq \emptyset$ for some non-trivial facet $\pi x \leq \pi_0$ of P_I. As $\pi x \leq \pi_0$ is generated by n affinely independent points of P_I, there exists an $\bar{x} \in P_I$ such that $\pi \bar{x} = \pi_0$ and

$\bar{x}_j = 1$ for some $j \in N^-$. (For, if not, then by the assumed affine independence we have that $|N^-| = 1$. It follows that $\pi_0 = 0$ and since all unit vectors of $\mathbf{R}^n$ are feasible points, that $\pi x \leq \pi_0$ is a *trivial* facet of the form $x_j \geq 0$). But the point $\bar{\bar{x}}$ given by $\bar{\bar{x}}_j = \bar{x}_j$, $j \in N - N^-$, $\bar{\bar{x}}_j = 0$, $j \in N^-$ is contained in P_I as A is non-negative and hence, $\pi\bar{\bar{x}} > \pi_0$ which is impossible. Consequently, every non-trivial facet $\pi x \leq \pi_0$ satisfies $\pi_j \geq 0$ for all $j \in N$ and $\pi_0 > 0$.

Denote by $S = S(\pi)$ the support of π, i.e. $S = \{j \in N \mid \pi_j > 0\}$. Let $G_S = (S, E_S)$ be the induced subgraph of G with node set S and edge set $E_S \subseteq E$ and let $P^S = P \cap \{x \in \mathbf{R}^n \mid x_j = 0$ for all $j \notin S\}$. Due to the non-negativity of A, one verifies readily that the convex hull P_I^S of integer points of P^S satisfies $P_I^S = P_I \cap \{x \in \mathbf{R}^n \mid x_j = 0$ for all $j \notin S\}$. Furthermore, $\pi x \leq \pi_0$ *retains its property* of being a facet of P_I^S. If one considers *proper* subgraphs of G_S, this property of $\pi x \leq \pi_0$ may or may not be "inherited". In fact, if all components of π are (non-negative) integers and $\pi_0 = 1$, then one readily verifies that for *all* subgraphs (including those on single nodes) the property of $\pi x \leq \pi_0$ to be a facet of the resulting (lower-dimensional) packing polyhedron is retained. The next theorem characterizes all facets of P_I with integer π and $\pi_0 = 1$, see [9, 18].

Theorem 1. *The inequality $\sum_{j \in K} x_j \leq 1$, where $K \subseteq N$, is a facet of P_I if and only if K is the node set of a clique (maximal complete subgraph) of G.*

Thus one knows *all* subgraphs of the intersection graph G of a zero-one matrix A that give rise to facets $\pi x \leq \pi_0$ with nonnegative integer π_j, $j = 1, \ldots, n$, and $\pi_0 = 1$. If, however, $\pi_0 \geq 2$ and π_0 does *not* divide all components of π, then there exists a *smallest* subgraph G' of G, G' not an isolated node, such that $\pi x \leq \pi_0$ looses its property of being a facet of the packing polytope associated with the packing problem of any *proper* subgraph of G'. If $\pi > 0$, i.e. if $S = N$, then, of course, the (full) intersection graph G may have this property and thus G may be itself *strongly facet-producing* [24].

Definition. A vertex-induced subgraph $G_S = (S, E_S)$ of $G = (N, E)$ with node set $S \subseteq N$ is *facet producing* if there exists an inequality $\pi x \leq \pi_0$ with nonnegative integer components π_j such that (i) $\pi x \leq \pi_0$ is a facet of $P_I^S = P_I \cap \{x \in \mathbf{R}^n \mid x_j = 0$ for all $j \notin S\}$ and (ii) $\pi x \leq \pi_0$ is *not* a facet for $P_I^T = P_I \cap \{x \in \mathbf{R}^n \mid x_j = 0$ for all $j \notin T\}$ where T is any subset of S such that $|T| = |S| - 1$. A subgraph G_S of G is called *strongly facet-producing* if there exists an inequality $\pi x \leq \pi_0$ such that (i) holds and (ii) holds for all $T \subset S$ satisfying $|T| < |S| - 1$. A subgraph G_S of G is *facet defining* if there exists an inequality $\pi x \leq \pi_0$ such that (i) holds and (ii) such that $\pi_j > 0$ for $j \in S$. (Shortly we will say that G_S defines the facet $\pi x \leq \pi_0$.)

Remark 1. Every strongly facet-producing (sub-)graph is facet-producing. Every facet-producing (sub-)graph is facet-defining. If G_S is facet-defining, then G_S is connected.

Proof. The first two parts being obvious, let $\pi x \leq \pi_0$ be a facet defined by $G_S = (S, E_S)$, i.e. $\pi_j > 0$ for $j \in S$, and suppose that G_S is not connected. Then $G_S = (S, E_S)$ can be written as $G_S = G_1 \cup G_2$ where $G_i = (S_i, E_i)$ with $S_i \neq \emptyset$ for $i = 1, 2$ and $S = S_1 \cup S_2$, $S_1 \cap S_2 = \emptyset$ and $E_S = E_1 \cup E_2$. Let P_I^i be defined like P_I^S with S replaced by S_i and let $\pi_0^i = \max\{\pi x \mid x \in P_I^i\}$ for $i = 1, 2$. Since $\pi x \leq \pi_0$ is defined by G_S, it follows that $\pi_0^i > 0$ for $i = 1, 2$. Define $\pi_j^i = \pi_j$ for $j \in S_i$, $\pi_j^i = 0$ for $j \notin S_i$ and let $x^i \in P_I^i$ be such that $\pi x^i = \pi_0^i$ for $i = 1, 2$. Since G_S is disconnected, $x^1 + x^2 \in P_I^S$ and hence, $\pi_0^1 + \pi_0^2 \leq \pi_0$. It follows that every $x \in P_I^S$ satisfying $\pi x = \pi_0$ satisfies $\pi^i x = \pi_0^i$ for $i = 1, 2$ and consequently, $\pi x \leq \pi_0$ is not a facet of P_I^S.

By the discussion preceding the definition, it is clear that every facet $\pi x \leq \pi_0$ of P_I is either produced by the subgraph G_S having $S = S(\pi)$ or if not, that there exists a subgraph G_T of G_S with $T \subseteq S$ such that the facet $\bar{\pi} x \leq \pi_0$ of P_I^T is produced by G_T where $\bar{\pi}_j = \pi_j$ for $j \in T$, $\bar{\pi}_j = 0$ for $j \notin T$ and P_I^T is defined as previously. The question is, of course, given $\bar{\pi} x \leq \pi_0$ can we retrieve the facet $\pi x \leq \pi_0$ of P_I. The answer is positive and follows easily from the following theorem which can be found in [16].

Theorem 2. *Let $P_I^S = P_I \cap \{x \in R^n \mid x_j = 0$ for all $j \notin S\}$ be the set packing polyhedron obtained from P_I by setting all variables x_j, $j \in N - S$, equal to zero. If the inequality $\sum_{j \in S} a_j x_j \leq a_0$ is a facet of P_I^S, then there exist integers β_j, $0 \leq \beta_j \leq a_0$, such that $\sum_{j \in S} a_j x_j + \sum_{j \in N-S} \beta_j x_j \leq a_0$ is a facet of P_I.*

Generalizations of Theorem 2 for more general polyhedra encountered in zero-one programming problems have been discussed in [1, 4, 13, 14, 19, 26, 28].

The apparent conclusion from this result — in view of the notion of the intersection graph discussed above — is that the problem of identifying part (or all) of the facets of a set packing polyhedron P_I is thus equivalent to the problem of identifying *all* those subgraphs of the intersection graph G associated with a given zero-one matrix A that are *facet-producing* in the sense defined above. (It should be noted that the "facet defining" property of (sub-)graphs is a considerably *weaker* property as in this case we require solely that the corresponding facet has *positive* coefficients. The choice of terminology may seem somewhat arbitrary, but the positivity of *all* components of a facet furnished by a facet-defining (sub-)graph is crucial in some of the arguments to follow.) One possible attack on the problem of finding a linear characterization of P_I is thus to "enumerate" all possible graphs that are facet-producing, a truly difficult task as we will show in the next section.

3. Facet-producing graphs

Let $G = (N, E)$ be any finite undirected graph having no loops. Denote by A_G the incidence matrix of all cliques of G (rows of A_G) versus the nodes of G

(columns of A_G). Let $N = \{1, \ldots, n\}$ and let $P_G = \{x \in \mathbb{R}^n \mid A_G x \le e_G, x \ge 0\}$ where $e_G = (1, \ldots, 1)$ is dimensioned compatibly with A_G. As before let P_I denote the convex hull of integer points of P_G, i.e. $P_I = \operatorname{conv}\{x \in P_G \mid x \text{ integer}\}$. We will not note explicitly the dependence of the respective polyhedra upon the graph G which may be taken as the intersection graph associated with some given zero-one matrix. The term of a facet-producing (facet-defining) graph is used here analogously with $S = N$. If x is a node (edge) of G, then by $G - \{x\}$ we will denote the graph obtained from G by deleting node x from G and all edges incident to x from G (by deleting the edge x, but no node from G). Denote by $\bar{G}$ the *complement* of G, i.e. $\bar{G} = (N, \mathscr{E}(N) - E)$ where $\mathscr{E}(N)$ is the set of *all* edges on n nodes. Every clique in G defines a stable (independent) node set (or node packing) in $\bar{G}$ and every maximal stable node set in G defines a clique in $\bar{G}$. Let $Q_G = \{x \in \mathbb{R}^n \mid B_G x \le d_G, x \ge 0\}$ where B_G is the incidence matrix of all cliques in $\bar{G}$ and $d_G^T = (1, \ldots, 1)$ is dimensioned compatibly. Furthermore, let $Q_I = \operatorname{conv}\{x \in Q_G \mid x \text{ integer}\}$. One verifies readily that Q_G is the *anti-blocker* of P_I and that P_G is the anti-blocker of Q_I, see [9, 20].

Before investigating special facet-producing graphs it is interesting to note the following proposition which substantially reduces the search for facet-producing graphs. Contrary to what one might expect intuitively, it *is not* necessarily true that the complement $\bar{G}$ of a facet producing graph G is again facet-producing. The graph in Fig. 1 shows an example of a facet-producing graph G whose complement $\bar{G}$ is not facet-producing. In fact, whereas the graph G of Fig. 1 produces the facet $\sum_{i=1}^{10} x_i \le 4$, its complement $\bar{G}$ does not produce any facet in the sense of the above definition: rather, every facet of the associated packing problem is obtained by "lifting" the facets produced by some *proper* subgraph of $\bar{G}$, as the clique-matrix of $\bar{G}$ is of rank 9.

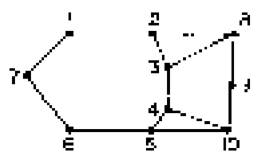


Fig. 1.

To state the next theorem we need the following definition from linear algebra: A square matrix M is said to be *reducible* if there exist permutation matrices P and Q such that

$$QMP = \begin{bmatrix} N & 0 \\ L & R \end{bmatrix} \tag{3.1}$$

where N and R are square matrices and 0 is a zero matrix. If no such permutation matrices exist, then M is called *irreducible*.

Theorem 3. *Suppose that G defines the facet $\pi x \le \pi_0$ for P_I such that*

$\max\{\pi x \mid x \in P_I\} = \pi\bar{x}$ is assumed at a vertex $\bar{x}$ of P_C satisfying $0 < \bar{x}_j < 1$ for $j = 1, \ldots, n$. If the submatrix A_1 of A_C for which $A_1\bar{x} = e_1$ is irreducible (and square), then the complement graph $\bar{G}$ is strongly facet producing.

Proof. Since every row of A_C defines a vertex of Q_I it follows from the assumption that $0 < \bar{x}_j < 1$ for all $j = 1, \ldots, n$ that the hyperplane $\bar{x}y = 1$ is generated by n linearly independent vertices of Q_I. On the other hand, $A_C\bar{x} \leq e_C$ implies validity of $\bar{x}y \leq 1$ for Q_I i.e. $Q_I \subseteq \{y \in \mathbb{R}^n \mid \bar{x}y \leq 1\}$. Hence, $\bar{G}$ defines the facet $\bar{x}y \leq 1$ of Q_I. Suppose that $\bar{x}y \leq 1$ defines a facet for some k-dimensional polyhedron $\bar{Q}_I \subseteq Q_I$ satisfying $k < n$. Then there exist k linearly independent vertices y^i of $\bar{Q}_I$ satisfying $\bar{x}y^i = 1$. Since the submatrix A_1 of A_C defining x is square, it follows that upon appropriate reordering of the rows and columns of A_1, A_1 can be brought into the form (3.1) contradicting the assumed irreducibility of A_1.

As an immediate consequence we have the corollary:

Corollary 3.1. *If G defines a facet $\pi x \leq \pi_0$ of P_I such that $\max\{\pi x \mid x \in P_C\} = \pi\bar{x}$ is assumed at a vertex $\bar{x}$ of P_C satisfying $0 < \bar{x}_j < 1$ for all $j = 1, \ldots, n$, then $\bar{G}$ defines a facet of Q_I.*

Chordless odd cycles ("holes") as well as their complements ("anti-holes") are known to define facets. In the former case one readily verifies the hypothesis of Theorem 3 to conclude that anti-holes as well as holes are strongly facet-producing. More recently, L. Trotter [24] has introduced the notion of a "web" which properly subsumes the aforementioned cases: A *web*, denoted $W(n, k)$ is a graph $G = (N, E)$ such that $|N| = n \geq 2$ and for all $i, j \in N$, $(i, j) \in E$ iff $j = i + k, i + k + 1, \ldots, i + n - k$, (where sums are taken modulo n), with $1 \leq k \leq [n/2]$. The web $W(n, k)$ is regular of degree $n - 2k + 1$, and has exactly n maximum node packings of size k. The complement $\bar{W}(n, k)$ of a web $W(n, k)$ is regular of degree $2(k - 1)$ and has exactly n maximum cliques of size k. One verifies that $W(n, 1)$ is a clique on n nodes, and for integer $s \geq 2$, $W(2s + 1, s)$ is an odd hole, while $W(2s + 1, 2)$ is an odd anti-hole. The following theorem is essentially from [24], see the appendix.

Theorem 4. *A web $W(n, k)$ strongly produces the facet $\sum_{j=1}^{n} x_j \leq k$ if and only if $k \geq 2$ and n and k are relatively prime. The complement $\bar{W}(n, k)$ of a facet-producing web $W(n, k)$ defines (strongly produces) the facet $\sum_{j=1}^{n} x_j \leq [n/k]$ if and only if $n = k[n/k] + 1$.*

The next theorem due to V. Chvátal [5] provides some graph-theoretical insights into graphs that give rise to facets with zero-one coefficients. To this end, recall that an edge e of a graph is called α-critical if $\alpha(G - e) = \alpha(G) + 1$, where $\alpha(G)$ denotes the stability number of G, i.e. the maximum number of independent nodes of G.

Theorem 5. *Let $G = (V, E)$ be a graph, let $E^* \subseteq E$ be the set of its α-critical edges. If $G^* = (V, E^*)$ is connected, then G defines the facet $\sum_{i \in V} x_i \le \alpha(G)$.*

It would be interesting to know whether all facets of set packing polyhedra having zero-one coefficients and a positive right-hand side constant can be described this way. (The question has been answered in the negative by Balas and Zemel [4a]). V. Chvátal also discusses in his paper [5] several graph-theoretical operations (such as the separation, join and sum of graphs) in terms of their polyhedral counterparts. Though very interesting in their own right, we will not review those results here. In particular, the two constructions given below are not subsumed by the graphical constructions considered by V. Chvátal.

We note next that graphs that satisfy the hypothesis of Theorem 5 need not be facet-producing in the sense defined in Section 2. In fact, the graph of Fig. 2 provides a point in-case. The facet defined by the graph G of Fig. 2 is given by $\sum_{i=1}^{6} x_i \le 2$ which, however, is produced by the odd cycle on nodes $\{1, 2, 3, 4, 5\}$. The coefficient of x_6 is obtained by "lifting" the facet $\sum_{i=1}^{5} x_i \le 2$, i.e. by applying Theorem 2.

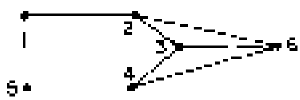


Fig. 2.

We next turn to a construction which permits one to "build" arbitrarily complex facet-producing graphs. Let G be any *facet defining* graph with node set $V = \{1, \dots, n\}$ with $n \ge 2$ and consider the graph G^* obtained by joining the ith node of G to the ith node of the "claw" $K_{1,n}$ by an edge. The claw $K_{1,n}$ — also referred to as a "cherry" or "star", see [13] — is the bipartite graph in which a single node is joined by n edges to n mutually non-adjacent nodes. We will give the node of $K_{1,n}$ that is joined to the ith node of G the number $n + i$ for $i = 1, \dots, n$, whereas the single node of $K_{1,n}$ that is not joined to any node of G, will be numbered $2n + 1$. (See Fig. 3 where the construction is carried out for a clique $G = K_4$.) Denote by $V^* = \{1, \dots, 2n + 1\}$ the node set of G^* and by E^* its edge-set. It turns out that G^* is facet-defining (this observation was also made by L. Woolsey [27]), and moreover, that G^* is strongly facet-producing.

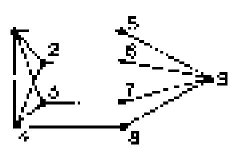


Fig. 3.

Theorem 6. *Let $G = (V, E)$ be a graph on $n \ge 2$ nodes and let $\pi x \le \pi_0$ be a (non-trivial) facet defined by G. Denote by $G^* = (V^*, E^*)$ the graph obtained from*

G by joining every node of G to the pending nodes of the claw $K_{1,n}$ as indicated above. Then G^* strongly produces the facet

$$\pi x^{(1)} + \pi x^{(2)} + \left(\sum_{i=1}^{n} \pi_i - \pi_0\right) x_{2n+1} \le \sum_{i=1}^{n} \pi_i \tag{3.2}$$

where $x^{(1)} = (x_1, \ldots, x_n)$, $x^{(2)} = (x_{n+1}, \ldots, x_{2n})$ and x_{2n+1} are the variables of the node-packing problem on G^* in the numbering defined above.

Proof. Let A_G be the clique-matrix of G and denote by P_I the set packing polyhedron defined with respect to A_G. The clique matrix of G^* is given by A_G^*:

$$A_G^* = \begin{bmatrix} A_G & 0 & 0 \\ I & I & 0 \\ 0 & I & e \end{bmatrix} \tag{3.3}$$

where I is the $n \times n$ identity matrix, e is vector with n components equal to one, and 0 are zero-matrices of appropriate dimension. Denote by P_I^* the set packing polyhedron defined with respect to A_G^*. To establish validity of (3.2) for P_I^* we note that $x_j = 1$ for some $j \in \{n+1, \ldots, 2n\}$ implies that $x_{2n+1} = 0$. Consequently, as $x^{(1)} + x^{(2)} \le e$, every vertex of P_I^* having $x_j = 1$ for some $j \in \{n+1, \ldots, 2n\}$ satisfies (3.2). On the other hand, since $\pi x^{(1)} \le \pi_0$ for every vertex of P_I^* and $\pi_0 < \sum_{i=1}^n \pi_i$ it follows that every vertex of P_I^* satisfies (3.2). To establish that the inequality (3.2) defines a facet of P_I^*, let B denote any $n \times n$ nonsingular matrix whose rows correspond to vertices of P_I satisfying $\pi x \le \pi_0$ with equality. Then define matrix B^* as follows:

$$B^* = \begin{bmatrix} B & 0 & e \\ I & E - I & 0 \\ 0 & e^T & 0 \end{bmatrix} \tag{3.4}$$

where I is the $n \times n$ identity matrix, E is the $n \times n$ matrix with all entries equal to one, e is the vector with n components equal to one, and 0 are zero-matrices of appropriate dimension. One verifies that the absolute value of the determinant of B^* is given by

$$\frac{1}{\pi_0}\left(\sum_{i=1}^{n} \pi_i\right) \det B.$$

Hence, B^* is non singular since $\det B \neq 0$ and $\sum_{i=1}^n \pi_i > \pi_0 > 0$. On the other hand, every row of B^* corresponds to some vertex of P_I^* satisfying (3.2) with equality. Consequently, (3.2) defines a facet of P_I^*. To prove that G^* produces a facet in the sense defined above, we show that no graph $G^* - \{j\}$ defines the facet (3.2) for $j = 1, \ldots, 2n + 1$. Note first that from the positivity of π it follows that every vertex of P_I^* satisfying (3.2) with equality satisfies $x_j + x_{n+j} + x_{2n+1} = 1$ for all $j \in V$. Let now $j \in V$ and consider $G^* - \{j\}$. Every vertex of P_I^* satisfying $x_j = 0$ and (3.2) with equality, necessarily satisfies $x_{n+j} + x_{2n+1} = 1$. Consequently, (3.2) does not define a facet of $P_j = P_I^* \cap \{x \in \mathbb{R}^{2n+1} \mid x_j = 0\}$ for $j \in V$. Consider next vertices of P_I^* satisfying (3.2) with equality and $x_{n+j} = 0$ for $j \in V$. Since $n \geq 2$ and $\pi x \leq \pi_0$ is *defined* by G, it follows that the node j of G has a neighbor $k(j)$ in G, i.e. $(j, k(j)) \in E$ for some $k(j) \neq j$, $k(j) \in V$. Consequently, every vertex satisfying (3.2) with equality and $x_{n+j} = 0$, also satisfies the equation $x_{n+k(j)} + x_{2n+1} = 1$. Hence (3.2) does not define a facet of P_{n+j} for $j \in V$. Finally, every vertex of P_I^* satisfying (3.2) with equality and $x_{2n+1} = 0$ satisfies the equation $x_k + x_{n+k} = 1$ for all $k \in V$, and consequently, (3.2) does not define a facet of P_{2n+1}. Consider next a subgraph $G' = (V', E')$ of G^* having $|V'| \leq 2n - 1$ nodes. If G' defines the facet given by (3.2), then, as noted earlier, all vertices of $P' = P_I^* \cap \{x \in \mathbb{R}^{2n+1} \mid x_j = 0, j \in V^* - V'\}$ satisfying (3.2) with equality must satisfy $x_j + x_{n+j} + x_{2n+1} \leq 1$ for all $j \in V$, since a vertex of P' is also a vertex of P_I^*. It follows that V' must contain the node numbered $2n + 1$ and furthermore, that $j \not\in V'$ implies $n + j \in V'$ for all $j \in V$. Consequently, $V \not\subseteq V'$ and $V' \cap \{n + 1, \ldots, 2n\} \neq \emptyset$. Let $N' \subseteq \{n + 1, \ldots, 2n\}$ be the nodes of G^* that are not in G'. Then either there exists a node $n + j \in N'$ such that the node $j \in V$ has a neighbor $k(j) \in V$ satisfying $n + k(j) \in V'$ or else, G is disconnected. The latter contradicts Remark 1. Consequently, by the above reasoning, we have $N' = \emptyset$, i.e. $V = V'$. This completes the proof of Theorem 6.

Corollary 6.1. *Let G, G^* and π be as in Theorem 6. If there exists a vertex $\bar{x} \in P_G$ such that $\max\{\pi x \mid x \in P_G\} = \pi\bar{x}$ is assumed at vertex $\bar{x}$ of P_G satisfying $0 < \bar{x}_j < 1$ for $j = 1, \ldots, n$, then the complement graph $\bar{G}^*$ of G^* defines a facet. Moreover, this facet of the set packing polyhedron Q_I^* associated with $\bar{G}^*$ (in rational form) is given by*

$$\bar{x} \cdot x^{(1)} + (e - \bar{x}) \cdot x^{(2)} + \bar{z} \cdot x_{2n+1} \leq 1 \tag{3.5}$$

where $\bar{z} = \min\{\bar{x}_j, j = 1, \ldots, n\}$.

Proof. Using the clique-matrix A_I^* as defined by (3.3) one verifies readily that the coefficients of the inequality (3.5) define a vertex of P_I^* with all components strictly between zero and one. Furthermore, the submatrix of A_I^* defining the vertex with components $(\bar{x}, e - \bar{x}, \bar{z})$ is nonsingular. As the cliques in G^* define vertices of Q_I^*, Corollary 6.1 follows.

The second construction uses edge-division. Let G be any facet-defining graph with node set $V = \{1, \ldots, n\}$ and edge-set E. Let $e = (u, w) \in E$ and consider the graph G^* with nodes set $V^* = \{1, \ldots, n, n + 1, n + 2\}$ and edge set

$$E^* = (E - \{e\}) \cup \{(u, n + 1), (n + 1, n + 2), (n + 2, w)\}.$$

That is, G^* is obtained from G by "inserting" two new nodes into an (existing) edge of G. Let $\pi x \leq \pi_0$ be the facet defined by G. As usual, we will assume that π is a vector of positive integers. An edge $e = (u, w) \in E$ will be called *π-critical* if there exists an independent node set F in the graph $G - \{e\}$ such that $\sum_{j \in F} \pi_j > \pi_0$ and $\sum_{j \in F - \{u\}} \pi_j = \pi_0$ or $\sum_{j \in F - \{w\}} \pi_j = \pi_0$. Note that π-criticality of an edge is entirely analogous to the concept of α-criticality used above.

Theorem 7. *Let $G = (V, E)$ be a graph on $n \geq 3$ nodes and let $\pi x \leq \pi_0$ be a facet defined by G.*

Denote by $G^* = (V^*, E^*)$ the graph on $n+2$ nodes obtained from G by inserting two nodes $n+1$ and $n+2$ into a π-critical edge $e = (v, w) \in E$. Then G^* defines the facet

$$\pi x + \pi_*(x_{n+1} + x_{n+2}) \leq \pi_0 + \pi_* \tag{3.6}$$

where $\pi_* = \min(\pi_v, \pi_w)$.

Proof. Denote by P_I the set packing polyhedron defined with respect to the clique matrix of G and let P_I^* be defined correspondingly with respect to G^*. Validity of the inequality (3.6) is immediate. Let B be any $n \times n$ nonsingular matrix of vertices of P_I that satisfy $\pi x \leq \pi_0$ with equality. Note that every vertex of P_I is a vertex of P_I^*. We show next that among the linearly independent vertices of P_I satisfying $\pi x \leq \pi_0$ with equality, there exists at least one vertex such that $x_v = x_w = 0$. For suppose not, then every vertex $\bar{x}$ of P_I such that $\pi\bar{x} = \pi_0$ satisfies $x_v + x_w = 1$. But by assumption, $\pi x \leq \pi_0$ has at least three non-zero components. Consequently, since $\pi x \leq \pi_0$ defines a facet of P_I, there exists a vertex with the asserted property. Consider the matrix B^* defined as follows

$$B^* = \begin{bmatrix} B & a & b \\ C & 0 & 1 \\ d & 0 & 0 \end{bmatrix} \tag{3.7}$$

where B is the $n \times n$ matrix defined above. The vector a has a 1 entry if in the associated row of B the component with number v is zero, zeros elsewhere. The vector b has a 1 entry if the corresponding component of a is zero and if in the associated row of B the component with number w is zero; zeros elsewhere. As there exists at least one row in B such that in both positions v and w there are zeros, we let C be a duplicate of that row. Finally, d is the incidence vector of the stable set F in $G - \{e\}$ for which $\sum_{j \in F} \pi_j > \pi_0$. Using standard linear algebra arguments, one verifies that B^* is nonsingular since, by construction, $CB^{-1}b = 0$, $a + b = e$ and $dB^{-1}e > 1$. Consequently, the inequality (3.6) defines a facet of P_I^*.

Note that edge-division does not always yield facet-producing graphs if the construction is used on facet-defining graphs. An example to this point is provided by the complete graph K_4 on the node set $\{1, 2, 3, 4\}$ and the inequality $\sum_{j=1}^{4} x_j \leq 1$ defined by K_4. If we insert two nodes 5 and 6 into the edge $\{3, 4\}$, the inequality (3.6) defined by G^* is produced by the odd hole on nodes $\{1, 3, 5, 6, 4\}$ whereas the coefficient of node 2 is obtained by "lifting" the inequality $x_1 + x_3 + x_5 + x_6 + x_4 \leq 2$. On the other hand, if the second construction is used on an odd hole on 5 nodes one obtains successively all odd holes. We thus suspect that G^* is (strongly) facet-producing if one assumes in Theorem 7 that G is (strongly) facet-producing rather than facet-defining.

To illustrate the foregoing, let us consider the graph G of Figure 5. The facet

$\pi x \le \pi_0$ produced by the graph is given by $\sum_{i=1}^{8} x_i + 3x_9 \le 4$. As one readily verifies, every edge of G is π-critical. Consequently, we can insert into any one of the edges of G two nodes; taking $e = (8,9)$ we get a new graph G^* and associated facet is $\sum_{i=1}^{8} x_i + 3x_9 + x_{10} + x_{11} \le 5$. Anyone of the edges of the graph G^* is again π-critical and we can continue inserting pairs of nodes into its edges, etc. Returning to the graph of Fig. 5 and adding a node 10 that is joined by edges to nodes 5, 6, 7, 8 and 9, we get from Theorem 2 the following facet *defined* (not produced) by the enlarged graph G': $\sum_{i=1}^{8} x_i + 3x_9 + 3x_{10} \le 4$. Upon inspection, we find that the (9, 10) of G' is π-critical. Inserting two nodes in the way described in Theorem 7 we obtain the facet defined by the resulting graph to be given by $\sum_{i=1}^{8} x_i + 3x_9 + 3x_{10} + 3x_{11} + 3x_{12} \le 7$. Using the construction of Theorem 6, we can get a fairly complex looking facet.

One might suspect from the foregoing that, given any set of positive integers $d_0, d_1, \ldots, d_n$ satisfying $d_i < d_0$ for $i = 1, \ldots, n$, at least four $d_i = 1$ and $\sum_{i=1}^{n} d_i > d_0$, there exists a graph G producing a facet $\pi x \le \pi_0$ such that $\pi_i = d_i$ for $i = 0, 1, \ldots, n$, *provided* that n is chosen sufficiently large. (The answer to this problem is definitely in the negative for small n.) My guess is that the answer is positive.

The foregoing may suggest that the complexity of the facial structure of set packing polyhedra renders useless pursuit of this line of research as regards its use in any computation utilizing linear programming relaxations. The following example may serve to indicate the contrary and points to an interesting question that, presumably, can only be answered in a statistical sense.

Example. Consider the maximum-cardinality node-packing problem on an odd anti-hole G with $n \ge 5$ vertices and let A_G denote the edge vs. node incidence matrix of G. Denote by R the following permutation matrix:

$$R = \begin{bmatrix} 0 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & & & & & \vdots \\ 0 & \ldots & \ldots & \ldots & 0 & 1 \\ 1 & 0 & \ldots & \ldots & \ldots & 0 \end{bmatrix}$$

We can write $A_G^T = (A_1^T, \ldots, A_p^T)$ where $p = [n/2] - 1$ and $A_i^T = (I + R^i)^T$ for $i = 1, \ldots, p$ with I being the $n \times n$ identity matrix. Let $P = \{x \in \mathbf{R}^n \mid A_G x \le e, x \ge 0\}$ be the linear programming relaxation of the node-packing problem and P_I the convex hull of integer solutions. As one readily verifies, $\max\{\sum_{i=1}^{n} x_i \mid x \in P\} = n/2$ for all n. But, the integer answer is *two* no matter what value n assumes, i.e. $\max\{\sum_{i=1}^{n} x_i \mid x \in P_I\} = 2$ for all n. Suppose now that we work with a linear programming relaxation of P_I utilizing a subset of the facets of P_I given in Theorem 1. Specifically, suppose that we have identified all cliques of G that are of maximum cardinality (this is in general a *proper* subset of *all* cliques of anti-holes). Denote by

$\bar{A}$ the corresponding clique-node incidence matrix. Then $\bar{A} = \sum_{i \neq h} R^i$. Let $\bar{P} = \{x \in \mathbb{R}^n \mid \bar{A}x \leq e, x \geq 0\}$ be the linear programming relaxation of the node-packing problem on G. Then $P_I \subseteq \bar{P} \subseteq P$. As one readily verifies, $\max\{\sum_{j=1}^n x_j \mid x \in \bar{P}\} = 2 + 1/\lfloor n/2 \rfloor$ and the integer optimum of 2 follows by simply rounding down.

The interesting fact exhibited by the example is that the knowledge of merely a few of the facets of P_I *in the case of odd anti-holes* permits one to obtain a bound on the integer optimum that is "sharp" as compared to the bound obtained by working on the linear programming relaxation involving the edge-node incidence matrix of the anti-hole (which is *arbitrarily bad* according to how large one chooses n). The general question raised by this example is of course, *how often* (in a statistical sense) it will be sufficient to work with only a small subset of all facets of a set packing polyhedron P_I (such as those given by cliques, holes, etc.) in order to verify ε-optimality of some extreme point of P_I with respect to some linear form cx, where ε is some given tolerance-level measuring the distance of an l.p. optimum from the true integer optimum objective function value.

Acknowledgement

I am indebted to E. Balas and L.E. Trotter, Jr. for helpful criticism of an earlier version of this paper. In particular, Les Trotter pointed out to me an error in the original proof of Theorem 6.

Appendix

As Theorem 4 asserts more than proven in [24], we shall provide a proof of the new part in Theorem 4, which states that the complement $\bar{W}(n,k)$ of a facet-producing web $W(n,k)$ strongly produces the facet $\sum_{j=1}^n x_j \leq h$ if and only if $n = kh + 1$, where $h = \lfloor n/k \rfloor$. We first prove the only-if part of the sentence. To do so, it suffices to show that the web $W(n,k)$ contains a (properly smaller) facet-producing web $W(n',k')$ with $\lfloor n'/k' \rfloor = h$ if $k \geq 2$, n and k are relatively prime and $n = kh + j$ with $2 \leq j \leq k-1$. Let $k' = \lfloor k/j \rfloor + 1$ and $n' = k'h + 1$. Obviously, $k' \geq 2$ and g.c.d. $(n', k') = 1$. To see that $W(n',k')$ is a (vertex induced) subgraph of $W(n,k)$, we check the necessary and sufficient conditions for containment of Theorem 4 of [24] which require that (i) $nk' \geq n'k$ and (ii) $n(k'-1) \leq n'(k-1)$. (i) follows because $\lfloor k/j \rfloor + 1 \geq k/j$. (ii) follows because $h(k - k') + k - 1 - j\lfloor k/j \rfloor \geq 0$. The latter holds because g.c.d. $(n,k) = 1$ implies $k - j\lfloor k/j \rfloor \geq 1$. Since $W(n',k')$ is contained in $W(n,k)$, the complement $\bar{W}(n,k)$ of $W(n,k)$ contains a subgraph defining the facet $\sum x_j \leq h$ where the summation extends over a proper subset of all vertices of $\bar{W}(n,k)$. Hence the facet $\sum_{j=1}^n x_j \leq h$ is not produced by $\bar{W}(n,k)$. To prove the if-part of the above sentence, we note that the vertex-sets $C_i = \{i, i+k, \ldots, i+(h-1)k\}$ define maximum cliques in $\bar{W}(n,k)$ where $i = 1, \ldots, n$ and indices are taken modulo n. Let B be the incidence matrix of these cliques and note that $BA^T = E - R$ where A is the incidence matrix of all cliques in $\bar{W}(n, k)$ (see [24]), E is a matrix of ones and R is a permutation matrix. To prove that B contains *all* maximum cliques of $W(n, k)$ let b be the incidence vector to any maximum clique of $W(n, k)$. Then $bB^{-1} = e - bA^TR^T \geq 0$ implies that $bx \leq 1$ is inessential in defining $P = \{x \in \mathbf{R}^n \mid Bx \leq e, x \geq 0\}$ or alternatively, identical to one of the rows of B. (The vector e is the vector of n ones.) Hence, since P contains the set-packing polyhedron associated with $W(n, k)$, B contains the incidence vectors of all maximum cliques of $W(n, k)$. Using an argument entirely analogous to the one used in the proof of Theorem 2 of [24], one shows that the matrix B is irreducible and hence, by Theorem 3, $\bar{W}(n, k)$ produces the facet $\sum_{i=1}^{n} x_i \leq h$ if $n = kh - 1$.

References

[1] E. Balas, Facets of the knapsack polytope, MSRR No. 323, Carnegie-Mellon University, September 1973. Forthcoming in *Math. Programming*.

[2] E. Balas and R. Jeroslow, Canonical cuts on the unit hypercube, *SIAM J. on Appl. Math.* 23 (1972) 61–69.

[3] E. Balas and M.W. Padberg, Set partitioning, in: B. Roy (ed.), *Combinatorial Programming: Methods and Applications* (Reidel Publishing Company, Dordrecht, 1975).

[4] E. Balas and E. Zemel, All the facets of the knapsack polytope, MSSR No. 374, Carnegie-Mellon University, 1975.

[4a] E. Balas and E. Zemel, Critical cutsets of graphs and canonical facets of set packing polytopes, MSSR No. 385, Carnegie-Mellon University, 1976.

[5] V. Chvátal, On certain polytopes associated with graphs, CRM-238, Université de Montréal, October 1973. Forthcoming in the *J. Comb. Theory*.

[6] J. Edmonds, Covers and packings in a family of sets, *Bull. Am. Math. Soc.* 68 (1962) 494–499.

[7] J. Edmonds, Paths, trees and flowers, *Canadian J. Math.* 17 (1965) 449–467.

[8] J. Edmonds, Maximum matching and a polyhedron with 0, 1 vertices, *J. Res. National Bureau of Standards* 69B (1965) 125–130.

[9] D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra, *Math. Programming* 1 (1971) 168–194.

[10] R. Garfinkel and G.L. Nemhauser, A survey of integer programming emphasizing computation and relations among models, in: T.C. Hu and S.M. Robinson, eds., *Mathematical Programming* (Academic Press, 1973).

[11] F. Granot and P.L. Hammer, On the use of boolean functions in 0–1 Programming, O.R. Mimeograph No. 70, Technion, 1970.

[12] P.L. Hammer, E.L. Johnson and U.N. Peled, Facets of regular 0–1 polytopes, CORR 73-19, University of Waterloo, October 1973.

[13] F. Harary, *Graph Theory* (Addison-Wesley, Reading, MA, 1969).

[14] E.L. Johnson, A class of facets of the master 0–1 knapsack polytope, Thomas J. Watson Research Center Report RC 5106, IBM Research, October 1974.

[15] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller et al., eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972).

[16] G.L. Nemhauser and L.E. Trotter, Properties of vertex packing and independence system polyhedra, *Math. Programming* 6 (1974) 48–61.

[17] M.W. Padberg, Essays in integer programming, Ph.D. Thesis, Carnegie Mellon University, May, 1971.

[18] M.W. Padberg, On the facial structure of set packing polyhedra, *Math. Programming* 5 (1973) 199–215.

[19] M.W. Padberg, A note on zero-one programming, *Operations Res.*, 23 (1975) 833–837.
[20] M.W. Padberg, Perfect zero-one matrices, *Math. Programming*, 6 (1974) 180–196.
[21] M.W. Padberg, Almost integral polyhedra related to certain combinatorial optimization problems, GBA Working Paper 75-25, 1975, New York University, forthcoming in *Linear Algebra and its Applications*.
[22] H.M. Salkin and J. Saha, Set covering: uses, algorithms, results, Technical Memorandum No. 272, Case Western Reserve University, March 1973.
[23] L.E. Trotter, Solution characteristics and algorithms for the vertex packing problem, Technical Report No. 168, Operations Research, Cornell University, 1973.
[24] L.E. Trotter, A class of facet producing graphs for vertex packing polyhedra, *Discrete Math.* 12 (1975) 373.
[25] H. Weyl, Elementare Theorie der konvexen Polyeder, *Comm. Math. Helv.* 7, 1935, 290–306 (translated in Contributions to the Theory of Games, Vol. I, 3–18, Annals of Mathematics Studies, No. 24, Princeton, 1950).
[26] L. Wolsey, Faces for linear inequalities in zero-one variables, CORE Discussion Paper No. 7338, November 1973.
[27] L. Wolsey, Oral Communication, Bonn, September 1975.
[28] E. Zemel, Lifting the facets of 0–1 polytopes, MSRR No. 354, Carnegie-Mellon University, December 1974.

Annals of Discrete Mathematics 1 (1977) 435–456

PROPERTIES OF FACETS OF BINARY POLYTOPES

Uri N. PELED

Department of Mathematics, University of Toronto, Toronto, Ont., Canada

Properties of facets of full dimensional polytopes P with binary vertices are studied. If Q is obtained from P by fixing some of the binary variables, then the facets of P that reduce to a given facet of Q are determined by the vertices of a certain polyhedron V. The case where V has a unique vertex is characterized. If P is completely monotonic and the facet of Q has 0-1 coefficients, then the vertices of V lie in a hypercube of side 1, and the integer vertices correspond to the sequential lifts or extensions. The self facets, i.e. hyperplanes spanned by binary points, are connected to the hyperplanes spanned by non-negative integral points. Every threshold function can be labelled by its Chow parameter vector. The faces of the convex hull of all n-argument parameter vectors are characterized. This leads to a necessary and sufficient condition for a parameter vector to label a self dual threshold function having a self facet separator.

1. Introduction

This paper deals with the facets of full-dimensional polytopes with binary vertices, i.e. the convex hulls of feasible solutions of binary programming problems. Section 2 is a unification and generalization of previous results by several authors on the connection between facets of such a problem P and the facets of a subproblem Q obtained by fixing some of the variables of P to binary values. The facets of P that reduce to a given facet of Q ("lift/extensions") are shown to be determined by the vertices of a certain polyhedron V, and the cases where V has only one vertex are characterized. Section 3 makes the further assumption that P is completely *monotonic* (a class that subsumes knapsack problems) and that the facet of Q has binary coefficients. The vertices of V are then shown to lie within a hypercube of side 1, and the integral vertices correspond precisely to the facets of P that can be obtained by "sequential" lifts or extensions. In Section 4 we examine the totality of facets of full-dimensional polytopes with binary vertices ("self facets"). They are shown to be connected to hyperplanes spanned by non-negative integral points. In Section 5 we reverse the point of view and ask what threshold functions have self facet "separators". Every threshold function (and some other Boolean functions) can be labelled by its Chow parameter vector. We characterize the non-empty faces of the convex hull of all n-argument Chow parameter vectors. The characterization of vertices and edges leads to a necessary and sufficient condition for a Chow parameter vector to label a self dual threshold function with a self facet separator.

2. Lifts and extensions

For an index set $N = \{1, \ldots, n\}$, let $S \subseteq B^N$ (B denotes the set $\{0, 1\}$) be a set of 0–1 N-vectors. Let N be partitioned into disjoint sets U, Z, F. Then by S^U_Z we mean the subset $T \subseteq B^F$ defined so that $x \in T$ if and only if the point y given by

$$y_j = \begin{cases} 1 & j \in U, \\ 0 & j \in Z, \\ x_j & j \in F, \end{cases}$$

is in S. Thus S^U_Z is obtained from S by fixing the components indexed by U and Z to 1 and 0, respectively, and then taking only the F components of the points of S satisfying these conditions. When U or Z are empty we use the short notation S_Z or S^U. If S is the set of feasible solutions of some 0–1 programming problem, or in short a *problem*, then S^U_Z corresponds to the *subproblem* obtained by fixing x_j, $j \in U \cup Z$ as above. An important class of problems is that of the monotone ones. S is *monotone* if whenever $x \in S$ and some components of x are changed from 1 to 0, the resulting point is still in S. In this section we relate the facets of conv (S) and conv (S^U_Z) (conv denotes convex hull).

A linear inequality is said to be *valid* for a set of points when it is satisfied by all points in the set, and to *support* the set if in addition some points of the set satisfy it with equality. Clearly an inequality is valid for (supports) a polytope if and only if it is valid for (supports) the set of its vertices (a *polytope* is a convex hull of a finite set of points).

Definition 1. Let S^U_Z be non-empty and let

$$\sum_{j \in F} a_j x_j \leq a_0 \tag{1}$$

be a valid inequality for S^U_Z. For each subset $Z' \subseteq Z$, the *extension coefficient* $e_{Z'}$ (of (1) relative to Z') is defined by

$$e_{Z'} = a_0 - \max_{x \in S^U_{Z-Z'}} \sum_{j \in F} a_j x_j, \tag{2}$$

where the maximum above is $-\infty$ if no x satisfies the condition. Similarly for each subset $U' \subseteq U$, the *lift coefficient* $l_{U'}$ is defined by

$$l_{U'} = \max_{x \in S^{U-U'}_Z} \sum_{j \in F} a_j x_j - a_0, \tag{3}$$

where the maximum above is $-\infty$ if no x satisfies the condition.

Proposition 1. *Let S be monotone, $S^U_Z \neq \emptyset$ and (1) valid for S^U_Z. Then*

(i) $e_{Z'} \geq 0$;

(2) *$l_{U'}$ is finite;*
(3) *if (1) supports S^U_Z, then $l_{U'} \ge 0$.*

Proof. By monotonicity $S^{U\cup Z'}_{Z-Z'} \subseteq S^U_Z$ and so

$$\max_{x\in S^{U\cup Z'}_{Z-Z'}} \sum_{j\in F} a_jx_j \le \max_{x\in S^U_Z} \sum_{j\in F} a_jx_j \le a_0,$$

which proves (1). Similarly $S^{U-U'}_{Z\cup U'} \supseteq S^U_Z \ne \emptyset$, and so

$$\max_{x\in S^{U-U'}_{Z\cup U'}} \sum_{j\in F} a_jx_j \ge \max_{x\in S^U_Z} \sum_{j\in F} a_jx_j$$

which proves (2). Moreover, if (1) supports S^U_Z the last right-hand side is a_0, which proves (3). $\square$

The extension and lift coefficients impose conditions on the coefficients of valid inequalities for S that reduce back to (1) under the substitution $x_j = 1$, $j\in U$, $x_j = 0$, $j\in Z$.

Proposition 2. *Let S^U_Z be non-empty and let (1) be valid for it. If the inequality*

$$\sum_{j\in N} a_jx_j \le a_0 + \sum_{j\in U} a_j \tag{4}$$

is valid for S, then for each $Z'\subseteq Z$, $\sum_{j\in Z'} a_j \le e_{Z'}$ and for each $U'\subseteq U$, $\sum_{j\in U'} a_j \ge l_{U'}$. In particular $a_j \le e_j$ for all $j\in Z$ and $a_j \ge l_j$ for all $j\in U$.

Proof. To prove $\sum_{j\in Z'} a_j \le e_{Z'}$, we may assume that $e_{Z'}$ is finite. Therefore there exists a point $x\in S^{U\cup Z'}_{Z-Z'}$ satisfying $e_{Z'} + \sum_{j\in F} a_jx_j = a_0$. But since $S^{U\cup Z'}_{Z-Z'}$ is a subproblem of S and (4) is valid for S, x must also satisfy $\sum_{j\in Z'} a_j + \sum_{j\in F} a_jx_j \le a_0$. The bound for lifts is proved similarly. $\square$

We can prove the converse of Proposition 2 for pure extensions ($U = \emptyset$) or pure lifts ($Z = \emptyset$).

Proposition 3. *If the inequality $\sum_{j\in N-Z} a_jx_j \le a_0$ is valid for S_Z, and if $\sum_{j\in Z'} a_j \le e_{Z'}$ holds for each $Z'\subseteq Z$, then $\sum_{j\in N} a_jx_j \le a_0$ is valid for S. Similarly if $\sum_{j\in N-U} a_jx_j \le a_0$ is valid for S^U, and if $\sum_{j\in U'} a_j \ge l_{U'}$ holds for each $U'\subseteq U$, then $\sum_{j\in N} a_jx_j \le a_0 + \sum_{j\in U} a_j$ is valid for S.*

Proof. Let $x\in S$ and let $Z' = \{j\in Z \mid x_j = 1\}$. By (2) we have $e_{Z'} + \sum_{j\in N-Z} a_jx_j \le a_0$, and since $e_{Z'} \ge \sum_{j\in Z'} a_j = \sum_{j\in Z} a_jx_j$, x satisfies $\sum_{j\in N} a_jx_j \le a_0$. The result for lifts has a similar proof. $\square$

The preceding discussion can be generalized to mixed lift/extensions. If S^U_Z is non-empty and (1) is valid for it, then for each $Z'\subseteq Z$, $U'\subseteq U$ we may define the coefficient

$$c_{Z'U'} = \max_{x \in S(U', Z')} \sum_{j \in T} a_j x_j .$$

It can then be shown that $c_{Z,\emptyset} = a_0 - e_Z$, $c_{\emptyset,U} = a_0 + f_U$ and that (4) is valid for S if and only if for each $Z' \subseteq Z$, $U' \subseteq U$,

$$c_{Z'U'} \leq a_0 - \sum_{j \in U'} a_j - \sum_{j \in Z'} a_j .$$

Let us now turn to examine conditions under which (4) is not only valid for S, but also a facet of conv (S). We recall that a *polyhedron* is the solution set of a finite number of linear inequalities. Bounded polyhedra are the same as polytopes. The *dimension* of a polyhedron P is one less than the maximum number of affinely independent points of P. A *face* of a polyhedron P is the solution set of the system obtained by replacing some of the inequalities defining P by equalities. In particular, vertices are 0-dimensional faces, *edges* are 1-dimensional faces and *facets* are faces of dimension one less than that of P. The faces of P are the same as the extreme subsets of P and also the sets of optimal solutions of linear programs over P. If P is *full-dimensional* (i.e. its dimension equals the number of variables), then in any system of linear inequalities defining P, the irredundant inequalities correspond precisely (up to proportion) to the facets of P. As is customary, we call these inequalities themselves the facets of P. To state the next result, we use the following definition.

Definition 2. Let (1) be a valid inequality for S_{Z}^{U}. Then its *valid polyhedron* is $V = \{a \in \mathbf{R}^{U \cup Z}$: every $x \in S$ satisfies (4)$\}$.

By definition, V is the polyhedron whose points are the U and Z components of all valid inequalities for S that reduce to (1) by the substitution $x_j = 1$, $j \in U$, $x_j = 0$, $j \in Z$. The remark following Proposition 3 gives a defining system for V in terms of $c_{Z'U'}$.

Proposition 4. *The valid polyhedron is full-dimensional and unbounded.*

Proof. If M is a large enough constant and

$$a_j = \begin{cases} -M & j \in Z \\ M & j \in U, \end{cases}$$

then $a \in V$. Thus V is not empty. Let d_j be the j unit vector. We show that if $a \in V$, then $a - d_j \in V$ for $j \in Z$ and $a - d_j \in V$ for $j \in U$. To prove the first of these statements please note that for all binary x, if $x_j = 0$, then (4) has the same form for $a - d_j$ as for a, and if $x_j = 1$, then (4) for $a - d_j$ is the sum of (4) for a and the valid inequality $-x_j \leq 0$. The second statement is proved similarly.

The next theorem belongs to the type of polarity results that are obtained by Aráoz [1] and also by Edmonds and Griffin [private communication].

Theorem 1. *Let* conv (S) *and* conv (S^U_Z) *be full-dimensional, and let* (1) *be a facet of the latter. Then* (4) *is a facet of the former if and only if* $a = (a_j) \in R^{|U \cup Z|}$ *is a vertex of the valid polyhedron* V *of* (1). *In that case* (4) *is called a lift/extension of* (1).

Proof. By definition, the validity of (4) means the same thing as $a \in V$. To show the "only if" part of the theorem, it is sufficient to prove that a is an extreme point of V. Suppose that $a = \frac{1}{2}(b + c)$, where $b, c \in V$. Then the two inequalities

$$\sum_{j \in F} a_j x_j + \sum_{j \in U \cup Z} b_j x_j \le a_0 + \sum_{j \in U} b_j \tag{5}$$

$$\sum_{j \in F} a_j x_j + \sum_{j \in U \cup Z} c_j x_j \le a_0 + \sum_{j \in U} c_j \tag{6}$$

are valid for conv(S) and (4) is their arithmetic mean. Since conv(S) is full-dimensional and (4) is one of its facets, it must coincide with (5) and (6), otherwise it is redundant. Thus $a = b = c$, proving that a is extreme in V.

We now show the "if" part. This time we show that there are $n = |N|$ affinely independent points of S satisfying (4) with equality, proving that it is an $(n-1)$-dimensional face. For ease of writing, let us reindex the variables so that $U = \{1, \ldots, r\}$, $Z = \{r+1, \ldots, r+s\}$, $F = \{r+s+1, \ldots, n\}$. Since a is a basic solution of the system of inequalities (4) for all $x \in S$, there exist $r+s$ points $x^1, \ldots, x^{r+s} \in S$ such that a satisfies the corresponding inequalities (4) as equalities and the coefficient matrix of $a_1, \ldots, a_{r+s}$ in these inequalities, namely

$$X = \left[\begin{array}{ccc|ccc} x_1^1 - 1 & \cdots & x_r^1 - 1 & x_{r+1}^1 & \cdots & x_{r+s}^1 \\ \vdots & & & & & \\ x_1^{r+s} - 1 & \cdots & x_r^{r+s} - 1 & x_{r+1}^{r+s} & \cdots & x_{r+s}^{r+s} \end{array}\right],$$

is non-singular. Also since (1) is a facet of the full-dimensional conv(S^U_Z), there exist $n - r - s$ affinely independent points $y^{r+s+1}, \ldots, y^n \in S^U_Z$ that satisfy (1) with equality. Let these points form the rows of the matrix

$$Y = \begin{bmatrix} y^{r+s+1}_{r+s+1} & \cdots & y^{r+s+1}_n \\ \vdots & & \\ y^n_{r+s+1} & \cdots & y^n_n \end{bmatrix}.$$

By definition of S^U_Z, the points $x^{r+s+1}, \ldots, x^n$ defined by

$$x^i_j = \begin{cases} 1 & j \in U, \\ 0 & j \in Z, \\ y^i_j & j \in F \end{cases}$$

belong to S. They too satisfy (4) with equality. It remains to show that the rows of the matrix

$$X^* = \left[\begin{array}{ccc|ccc|ccc} x_1^1 & \cdots & x_r^1 & x_{r+1}^1 & \cdots & x_{r+s}^1 & x_{r+s+1}^1 & \cdots & x_n^1 \\ \vdots & & & \vdots & & & \vdots & & \\ x_1^{r+s} & \cdots & x_r^{r+s} & x_{r+1}^{r+s} & \cdots & x_{r+s}^{r+s} & x_{r+s+1}^{r+s} & \cdots & x_n^{r+s} \\ \hline & & & & & & y_{r+s+1}^{r+s+1} & \cdots & y_n^{r+s+1} \\ & 1 & & & 0 & & \vdots & & \\ & & & & & & y_{r+s+1}^{n} & \cdots & y_n^{n} \end{array}\right]$$

are affinely independent. If $r+s=n$, then the rows of X^* differ from the rows of X by a fixed translation $(1,\ldots,1,0,\ldots,0)$. As the rows of X are affinely independent, so are the rows of X^*. If $r+s<n$, subtract the last row of X^* from each other row. It is enough to show that the first $n-1$ rows are now linearly independent. These rows now constitute a matrix of the form $[\begin{smallmatrix} X & * \\ 0 & Y_0 \end{smallmatrix}]$, where the rows of X are linearly independent and the rows of Y_0 being the differences $y^{r+s+1}-y^n, \ldots, y^{n-1}-y^n$, are also linearly independent. This completes the proof of Theorem 1. □

Under the conditions of Theorem 1, suppose further that Z contains an index i such that the extension coefficient e_i relative to (1) is finite. If we consider S_Z^U as a subproblem of S_{Z-i}^U, then the valid polyhedron V is 1-dimensional with a vertex at e_i. Thus the inequality $\sum_{j \in T} a_j x_j - e_i x_i \leq a_0$ is a facet of conv(S_{Z-i}^U). If $Z-i$ contains a further index whose extension coefficient relative to the present inequality is finite, the process can be continued. This is called *sequential extension* of (1). In particular, if S is full-dimensional and monotone, so are all its subproblems of the form S_Z, and by Proposition 1 each facet of such a subproblem can be sequentially extended to (one or more, depending on the order of extension) facets of the complete problem. Hence, by Theorem 1, V has in fact vertices in that case. In a similar way one also has *sequential lifts* and *sequential lift/extensions*. Sequential extensions have been studied by many authors, including Balas [2], Balas and Zemel [3], Hammer, Johnson and Peled [7], Nemhauser and Trotter [12], Padberg [13], Pollatschek [15], Trotter [16], Wolsey [19] and Zemel [21]. Sequential lifts are treated by Wolsey [20], in a work that stimulated my interest in lifts. Theorem 1 was proved by Zemel [21] for the case of pure extensions. Non-sequential extensions are also discussed by Balas and Zemel [3].

We conclude this section with two corollaries and an example of Theorem 1.

Corollary 1. *Under the conditions of Theorem 1, if* $|U \cup Z| \leq 2$, *then every vertex of the valid polyhedron* V *corresponds to a sequential lift/extension of* (1).

Proof. We have already considered the case $|U \cup Z| = 1$. For $|U \cup Z| = 2$, consider the typical case of pure extensions, $U = \emptyset$, $Z = \{1, 2\}$, other cases being similar. We then have $V = \{(a_1, a_2) \mid a_1 \le e_1, a_2 \le e_2, a_1 + a_2 \le e_{12}\}$. It is easy to verify that if $e_1 + e_2 \le e_{12}$, then V has a unique vertex (e_1, e_2), and if $e_1 + e_2 > e_{12}$, then V has two vertices $(e_1, e_{12} - e_1)$ and $(e_{12} - e_2, e_2)$. All these vertices represent sequential extensions: in the first case the two sequences commute (give the same face) and in the second case they do not. □

Corollary 1 appears, for pure extensions, in Hammer, Johnson and Peled [7] and in Zemel [21].

Corollary 2. *Under the conditions of Theorem 1, assume further that all e_j, $j \in Z$ and l_j, $j \in U$ are finite. Then the following two conditions are equivalent:*

(1) *the inequality*

$$\sum_{j \in P} a_j x_j - \sum_{j \in Z} e_j x_j + \sum_{j \in U} l_j x_j \le a_0 + \sum_{j \in U} l_j \tag{7}$$

is valid for S;

(2) *the valid polyhedron V has a unique vertex (i.e. (1) has a unique lift/extension).*

In that case the unique vertex is in fact $(e_j, j \in Z;\ l_j, j \in U)$.

Proof. Please note that as e_j and l_j are finite, V has vertices (it being contained in an orthant of $\mathbf{R}^{U \cup Z}$).

(1) ⟹ (2). It is enough to show that whenever (4) is a facet of conv(S), $a_j = e_j$ for $j \in Z$ and $a_j = l_j$ for $j \in U$ (this follows from Theorem 1). Since (4) is valid for S, Proposition 2 gives $a_j \le e_j$ for $j \in Z$ and $a_j \ge l_j$ for $j \in U$. Therefore we can add the valid inequalities

$$(a_j - e_j)x_j \le 0 \qquad j \in Z,$$

$$(a_j - l_j)x_j \le a_j - l_j \qquad j \in U$$

to the valid inequality (7) to obtain (4). But (4) is a facet of the full dimensional conv(S), and so it is irredundant. It follows therefore that $a_j = e_j$ for $j \in Z$ and $a_j = l_j$ for $j \in U$.

(2) ⟹ (1). Note that by Theorem 1 there is a unique facet of conv(S) of the form (4). On the other hand, such facets can be obtained by sequential lift/extensions. The sequence may start from any $j \in U \cup Z$, since the lift/extension coefficients are all finite. This yields $a_j = l_j$ if $j \in U$ and $a_j = e_j$ if $j \in Z$. Therefore (7) is the unique facet in question and (1) certainly holds. □

Special cases of Corollary 2, involving pure extensions, appear in Balas [2], Hammer, Johnson and Peled [7] and Balas and Zemel [3].

Example. Let S be the set of incidence vectors of the node packings of the pentagon, i.e. the vectors $x \in B^5$ such that $x_i + x_{i+1} \le 1$ (indices modulo 5). S is full-dimensional and monotonic. The subproblem S^1 is not full-dimensional, because $x_1 = 1$ implies $x_2 = x_5 = 0$. Therefore let us consider the full-dimensional subproblem $S^1_{...} = \{x \in B^2 \mid x_1 + x_2 \le 1\}$ and the facet $x_1 + x_2 \le 1$ of its convex hull. The valid polyhedron V consists of $(\alpha_3, \alpha_4, \alpha_5)$ that satisfy

$$1 \le 1 + \alpha_5, \qquad 1 + \alpha_3 \le 1 + \alpha_4, \qquad 1 + \alpha_4 \le 1 + \alpha_5, \qquad \alpha_3 + \alpha_5 \le 1 + \alpha_4.$$

These inequalities are determined by the node packings $(1,0,0,0,0)$, $(1,0,1,0,0)$, $(0,1,0,0,1)$ and $(0,0,1,0,1)$, respectively. The other node packings give redundant inequalities. V is 3-dimensional and has two vertices $(0,0,0)$ and $(1,1,1)$. Thus the lift/extensions of $x_1 + x_2 \le 1$ are $x_1 + x_2 \le 1$ and $x_1 + x_2 + x_3 + x_4 + x_5 \le 2$. The second of these facets is the one "produced" by the pentagon in the sense of Trotter [16], i.e. it is not a pure extension of any subproblem. We see that it is not produced by the pentagon if we allow lift/extensions. It is not a sequential lift/extension (since $\alpha_3 = \alpha_5 = \alpha_4 = 0$). The first facet is a sequential lift/extension (lift in x_4 and then extend in x_3, x_5).

3. Completely monotonic problems

If (1) is valid for S^Z_U and (4) is valid for S, Proposition 2 gives upper bounds for α_j, $j \in Z$ and lower bounds for α_j, $j \in U$, namely the extension and lift coefficients, respectively. Under suitable conditions there are sharp opposite bounds for the α_j. We discuss here such conditions.

Definition 3. Let L and M be disjoint subsets of the index set N and let $S \subseteq B^N$. Then we write $L \succcurlyeq M$ (relative to S) when $S^L_M \subseteq S^M_L$. In that case L and M are said to be *comparable*. When $L \succcurlyeq M$ holds but $M \succcurlyeq L$ does not, we write $L \succ M$.

Informally, $L \succcurlyeq M$ means that if $x \in S$ and $x_j = 1$ for all $j \in L$, $x_j = 0$ for all $j \in M$, then by moving the ones from L to M we transform x into another point of S. As an example, consider the linear inequality

$$\sum_{j \in N} d_j x_j \le d_0 \tag{8}$$

and let $S = \{x \in B^N \mid x$ satisfies $(8)\}$. Such an S is called a *threshold set* or a *knapsack problem*, and the inequality (8) is a *separator* of S. Relative to this S we have $L \succcurlyeq M$ iff $\sum_{j \in L} d_j \ge \sum_{j \in M} d_j$, hence all disjoint sets are comparable.

The following properties of $\succcurlyeq$ are easily proved.

(1) If K, L and M are disjoint in pairs and $K \succcurlyeq L \succcurlyeq M$, then $K \succcurlyeq M$. This is true in particular for singletons.

(2) Every subproblem of S inherits from S the relations $\succcurlyeq$ between sets of its own variables. In other words, if $L, M \subseteq F$ and $L \succcurlyeq M$ relative to S, then $L \succcurlyeq M$ relative to S^E_F.

(ii) S is monotone if and only if every $i \in N$ satisfies $\{i\} \succeq \emptyset$. Thus if every singleton is comparable with $\emptyset$, S can be made monotone by complementing all variables x_i such that $\emptyset \succ \{i\}$.

Definition 4. S is *completely monotonic* if every two disjoint sets are comparable relative to S.

Winder [18] has shown that in order to establish complete monotonicity, it is sufficient to check only disjoint sets L and M such that $|L \cup M| \le \frac{1}{2}|N|$. As we have just seen, every threshold set S is completely monotonic. In order to discuss the converse statement, call S k-*summable* if for some $j = 2, \dots, k$ there exist j points $x^1, \dots, x^j \in S$ and j points $y^1, \dots, y^j \in B^n - S$ satisfying $x^1 + \cdots + x^j = y^1 + \cdots + y^j$. Otherwise S is k-*asummable*. It was shown by Elgot [6] that complete monotonicity is equivalent to 2-asummability, whereas the threshold property is equivalent to the property of k-asummability for every $k = 2, 3, \dots$. In fact, Winder [18] has shown that for every fixed k there are k-asummable sets that are not threshold. Thus the class of threshold sets is properly included in the class of completely monotonic sets.

If (4) is a valid inequality for S, a coefficient a_j, $j \in U$ is said to be *minimal* in (4) if any decrease in a_j makes (4) invalid. This is equivalent to the existence of a point $x \in S$ with $x_j = 0$ that satisfies (4) as an equality. For example, if (4) is a facet of conv(S), other than $x_j \le 1$, then a_j is minimal in (4). With these definitions we can now state the next result.

Theorem 2. *Let S be completely monotonic and monotone. Let*

$$\sum_{j \in J} x_j \le b, \quad J \subseteq N - U$$

be an inequality with 0–1 coefficients supporting S^U, and let

$$\sum_{j \in J} x_j + \sum_{j \in U} a_j x_j \le b + \sum_{j \in U} a_j \tag{10}$$

be valid for S. If, for some $i \in U$, a_i is minimal in (10), then $a_i \le l_i + 1$, where l_i is the lift coefficient of x_i in (9).

Proof. As remarked above, the relation $\succeq$ induces a total order on all the singletons. For ease of writing, let us reindex the variables so that $J = \{1, 2, \dots, |J|\}$ with

$$\{|J|\} \succeq \cdots \succeq \{2\} \succeq \{1\}. \tag{11}$$

Since (9) supports S^U, there exists a point of S^U that satisfies (9) with equality, i.e. has exactly b components from J equal to 1. By monotonicity of S^U we may take all the components outside J to be 0, and by (11) it follows that the point $x \in B^{N-U}$ given by

$$x_j = \begin{cases} 1 & j = 1,\dots,h \\ 0 & j = h+1,\dots,|J| \quad \text{or} \quad j \in N - U - J \end{cases}$$

belongs to S^U. A reformulation of this statement is that the point $x^i \in R^{N-U+i}$ given by

$$x^i_j = \begin{cases} 1 & j = 1,\dots,h \quad \text{or} \quad j = i \\ 0 & j = h+1,\dots,|J| \quad \text{or} \quad j \in N - U - J \end{cases} \tag{12}$$

belongs to S^{U-i}.

By (3) the lift coefficient l_i in (9) is given by

$$h + l_i = \max_{y \in S^{U-i}} \sum_{j \in J} y_j$$

and is finite by Proposition 1. For the same reasons as above, the optimal solution y can be taken to be the point

$$y_j = \begin{cases} 1 & j = 1,\dots,h+l_i \\ 0 & j = h+l_i+1,\dots,|J| \quad \text{or} \quad j \in N - U - J \end{cases}$$

The optimality of y implies that the point $y^i \in R^{N-U+i}$ given by

$$y^i_j = \begin{cases} 1 & j = 1,\dots,h+l_i+1 \\ 0 & j = h+l_i+2,\dots,|J| \quad \text{or} \quad j = i \quad \text{or} \quad j \in N - U - J \end{cases} \tag{13}$$

does not belong to S^{U-i}. Comparing (12) with (13) we see that x^i and y^i differ only at the components i and $h+1,\dots,h+l_i+1$. By complete monotonicity of S^{U-i}, the sets $\{i\}$ and $\{h+1,\dots,h+l_i+1\}$ must be comparable, and since x^i is in S^{U-i} and y^i is not, we conclude that

$$\{h+1,\dots,h+l_i+1\} \succ \{i\} \tag{14}$$

relative to S^{U-i}, hence relative to S too.

Let us now turn to the valid inequality (10), in which a_i was assumed to be minimal. This means that there exists a point $z \in S$ with $z_i = 1$ that satisfies (10) as an equality. For the same reasons as above we may assume that for some integer $k = 0, 1, \dots, |J|$, z satisfies $z_1 = \cdots = z_k = 1$ and $z_{k+1} = \cdots = z_{|J|} = 0$. Denoting $M = \{j \in U - i \mid z_j = 1\}$, we may then write the equality (10) in the form

$$k = h - a_i - \sum_{j \in M} a_j. \tag{15}$$

Since $a_i \geq l_i$ by Proposition 2, (15) yields

$$k \leq h - l_i - \sum_{j \in M} a_j. \tag{16}$$

We now distinguish two cases according to $\sum_{j\in M} a_j$.

Case 1: $\sum_{j\in M} a_j = 0$. By Propositions 1 and 2 we have $0 = \sum_{j\in M} a_j \geq l_M \geq 0$ and so $l_M = 0$. Therefore the maximum of $\sum_{j\in U} x_j$ is b not only for $x \in S^U$, but also for $x \in S^{U\cup M}$, and it follows that $\{b+1\} \succ M$. We claim that $k \leq b + l_i + 1$. Indeed, suppose that $k \geq b + l_i + 2$ were true. Since $\{b+1\} \succ M$ and since $z \in S$, the point u given by

$$u_j = \begin{cases} 0 & j = b+1 \\ 1 & j \in M \\ z_j & j \in N - M - \{b+1\} \end{cases}$$

belongs to S. But $u_i = z_i = 0$, by definition of M $u_j = 1$ for all $j \in U - i$, and $\sum_{j\in U} u_j = k - 1 \geq b + l_i + 1$. This contradicts the definition of the lift coefficient l_i and establishes the claim. But this claim, (15) and the condition of case 1 yield the desired result $a_i \leq l_i + 1$.

Case 2: $\sum_{j\in M} a_j \neq 0$. As in case 1 we have $\sum_{j\in M} a_j \geq 0$, hence $\sum_{j\in M} a_j > 0$. Since k is an integer, (16) yields

$$k \geq b + l_i + 1. \tag{17}$$

Let v be given by

$$v_j = \begin{cases} 1 & j = i \\ 0 & j = k - l_i, \dots, k \\ z_j & j \in N - \{i\} - \{k - l_i, \dots, k\}. \end{cases}$$

By (17), (11) and (14) we have $\{k - l_i, \dots, k\} \succ \{b+1, \dots, b + l_i + 1\} \succ \{i\}$, and therefore $v \in S$. Hence v satisfies the valid inequality (10), which reads $k - l_i - 1 \leq b + \sum_{j\in M} a_j$. This and (15) gives $a_i \leq l_i + 1$ and completes the proof of Theorem 2. □

A result analogous to **Theorem 2** holds for extensions instead of lifts. It assumes that a coefficient in a valid inequality is maximal rather than minimal. The theorem can be proved by methods close to and somewhat simpler than the ones for Theorem 2.

Theorem 3. *Let S be completely monotonic and monotone. Let*

$$\sum_{j\in T} x_j \leq b, \qquad T \subseteq N - Z \tag{18}$$

be an inequality with 0–1 *coefficients supporting S_Z, and let*

$$\sum_{j\in T} x_j + \sum_{j\in Z} a_j x_j \leq b \tag{19}$$

be valid for S. If, for some $i \in Z$, a_i is maximal in (19), then $a_i = e_i - 1$, where e_i is the extension coefficient of x_i in (18).

An important special case where the assumptions of Theorem 3 hold is when S is full-dimensional, J is a minimal set whose incidence vector is not in S (a "prime implicant" or a "minimal cover" of $\bar{S}$), Z is taken as $N - J$ and b is $|J| - 1$. Then (18) is a facet of the full-dimensional conv (S_Z). If (19) is a facet of conv (S), i.e. an extension of (18), then Theorem 3 applies to every $i \in Z$. This case was proved, for threshold sets S, by Balas and Zemel [3].

The following corollaries of the preceding two theorems assume that (9) (or (18)) is a facet of the full dimensional conv (S^U) (conv (S_Z)). Such facets with 0–1 coefficients have been characterized by Balas [2], Hammer, Johnson and Peled [7] and Wolsey [19]. The characterization has to assume no more than that S is "regular" (essentially that the singletons are comparable by $\succeq$) and this is covered anyhow by the stronger assumption that S is completely monotonic. The result is a characterization of the sequential lifts (extensions) of (9) (or (18)).

Corollary 3. *Let S be completely monotonic and monotone and let conv (S) and conv (S^U) be full-dimensional with the facets (10) and (9), respectively. Then (10) is a sequential lift of (9) if and only if $a = (a_j\ j \in U)$ is integral.*

Proof. The "only if" part is obvious, since sequential lifts of an inequality with integral coefficients have integral coefficients. To prove the "if" part, observe that by Proposition 2 and Theorem 2 each $j \in U$ satisfies $a_j = l_j$ or $a_j = l_j + 1$. Let $L = \{j \in U \mid a_j = l_j\}$ and $M = U - L$. Since (10) is valid for S, the inequality

$$\sum_{j \in F} x_j + \sum_{j \in L} l_j x_j \le b + \sum_{j \in L} l_j \tag{20}$$

is valid for S^M. By Corollary 1 and Theorem 1, (20) is the unique lift of (9) that is a facet of conv (S^M). Therefore this is a sequential lift of (9). It can be sequentially lifted further in M. The resulting coefficients will be integers, and since the resulting inequality will be a lift of (9) and a facet of conv (S), these coefficients must lie between l_j and $l_j + 1$, i.e. they are either l_j or $l_j + 1$. But none of these coefficients can be l_j, or else the facet (10) will be the sum of two valid inequalities. Therefore the resulting sequential lift is identical with (10). $\square$

The analogous result for extensions is expressed by

Corollary 4. *Let S be completely monotonic and monotone and let conv (S) and conv (S_Z) be full-dimensional with the facets (19) and (18), respectively. Then (19) is a sequential extension of (18) if and only if $a = (a_j\ j \in Z)$ is integral.*

This result was obtained by Balas and Zemel [3] under the conditions discussed above.

4. Self facets

In the previous sections we examined the question of how to obtain facets of a given problem from facets of a given subproblem. Here we look at the totality of facets of all full-dimensional polytopes with 0–1 vertices. Clearly if

$$\sum_{j\in N} a_j x_j \le a_0 \tag{21}$$

is such a facet, then it is also a facet of $\operatorname{conv}(S)$, where S is the threshold set $S=\{x\in B^n \mid x \text{ satisfies } (21)\}$. For this reason a hyperplane

$$\sum_{j\in N} a_j x_j = a_0 \tag{22}$$

is called a *self facet* when it is spanned by 0–1 points, i.e. when there are $|N|$ affinely independent points $x\in B^n$ satisfying (22). In studying the self facets, we do not lose generality by assuming that $a_1,\dots,a_n>0$. Not every threshold set S has a separator that is a facet of $\operatorname{conv}(S)$. For example, if $S=\{x\in B^3 : 2x_1+x_2+x_3\le 2\}$, no facet of $\operatorname{conv}(S)$ is a separator of S.

Given a hyperplane (22), let $\{h_i, i\in M\}$ be the set of distinct values among a_j, $j\in N$, and put $N_i=\{j\in N \mid a_j=h_i\}$, $i\in M$. Let $A:\mathbf{R}^N\to\mathbf{R}^M$ be a linear transformation defined by

$$(Ax)_i=\sum_{j\in N_i} x_j, \quad i\in M. \tag{23}$$

Clearly if $x\in B^n$ satisfies (22), then $t=Ax$ is an M-vector satisfying

$$\sum_{i\in M} h_i t_i = a_0, \tag{24}$$

$$0\le t_i\le |N_i|, \quad t_i \text{ integer}, \ i\in M. \tag{25}$$

Conversely, if t satisfies (25) and (24), then there is an x satisfying (22) and $t=Ax$. This correspondence carries over to facets as follows.

Proposition 5. *If (22) is a self facet then*
(1) (24) is spanned by points t satisfying (25);
(2) if $|N_i|\ge 2$ there is an integral t satisfying (24) and $0<t_i<|N_i|$.

Proof. Let X be a matrix whose rows are all the binary solutions of (22). Its columns are linearly independent. The rows of $T=AX$ satisfy (24) and (25). We claim that its columns are linearly independent and hence (1) holds. Indeed let $c\in\mathbf{R}^M$ satisfy $Tc=0$, and define $d\in\mathbf{R}^N$ by $d_j=c_i$ for $j\in N_i$. Then $Xd=0$, hence $d=0$, hence $c=0$. If (2) fails, then the i column of T consists solely of 0's and $|N_i|$'s. Hence all the columns of X indexed by N_i are equal to each other. Since $|N_i|\ge 2$, this contradicts the linear independence of the columns of X. □

Proposition 6. *Let (24) be spanned by non-negative integral points t. Then there exist numbers $(a_j, j \in N)$ such that (22) is a self facet and the sets of distinct values among $(b_i, i \in M)$ and among $(a_j, j \in N)$ are the same. In particular, if $(b_i, i \in M)$ are distinct, then $(a_j, j \in N)$ is obtained by duplication of the b_i.*

Proof. Let the rows of T be all the non-negative integral solutions of (24), with n_i' the largest entry in the i column of T. Put

$$n_i = \begin{cases} n_i' - 1 & \text{if } n_i' \geq 2 \text{ and no row } t \text{ of } T \\ & \text{satisfies } 0 < t_i < n_i', \\ n_i' & \text{otherwise.} \end{cases} \tag{26}$$

and define $N_i = \{n_1 + \cdots + n_{i-1} + 1, \ldots, n_1 + \cdots + n_i\}$, $N = \bigcup_{i \in M} N_i$. Then if $a_j = b_i$ for all $j \in N_i$ and the rows of X are the binary solutions of (22), we shall prove that the columns of X are linearly independent. Suppose that $Xd = 0$ for some $d \in \mathbf{R}^N$. We claim that for each $i \in M$, all the d_j, $j \in N_i$ are equal to each other. This is trivial for $n_i = 1$. For $n_i \geq 2$, (26) shows that there exists a row t of T satisfying $0 < t_i < n_i$. Since T is a submatrix of AX, where A is given by (23), there exists a row x of X such that exactly t_i of the x_j, $j \in N_i$ are equal to 1. All the $\binom{n_i}{t_i}$ different row vectors obtained from x by permuting the N_i components are also rows of X, and so are orthogonal to d. Hence by subtracting these equations from each other we establish the claim. Now it is possible to define $c \in \mathbf{R}^M$ by letting c_i be the common value of the d_j, $j \in N_i$, and hence $(AX)c = 0$, $Tc = 0$, $c = 0$ and $d = 0$. □

We give some examples illustrating the preceding propositions.

Examples. (1) If (24) has the form $2t_1 + 3t_2 = 12$, then the matrix

$$T = \begin{bmatrix} 6 & 0 \\ 3 & 2 \\ 0 & 4 \end{bmatrix}$$

has rank 2. The proof of Proposition 6 constructs the self facet (22) given by $a = (2,2,2,2,2,2,3,3,3,3)$, $a_0 = 12$. By inspection one can see that $a = (2,2,2,2,3,3,3,3)$ also gives a self facet, but $a = (2,2,2,3,3,3,3)$ does not.

(2) An example similar to the following one was shown to me by J.F. Maurras. The hyperplane $t_1 + 2t_2 + \cdots + 2^{k-1}t_k = 2^k$ is spanned by non-negative integral $t \in \mathbf{R}^k$. Indeed, the rows of the non-singular k by k matrix

$$\begin{bmatrix} 2 & 1 & 1 \cdots 1 \\ 0 & 2 & 1 \cdots 1 \\ & \ddots & \\ 0 & 0 & 0 \cdots 2 \end{bmatrix}$$

are solutions. In analogy with the proof of Proposition 6 we can construct the a_j as

$1, 2, 2, \ldots, 2^k$. Hence there is a self facet with $\sum n_i = 2k+1$ variables whose right hand side is 2^k, namely $x_0 + (x_1 + x_2) + 2(x_3 + x_4) + \cdots + 2^{k-1}(x_{2k-1} + x_{2k}) = 2^k$. This is a class of self facets where the largest coefficient is exponential in the number of variables.

(3) Given an arbitrary set $\{h_i, i \in M\}$ of positive integers, let a_0 be the least common multiple of the h_i. Then the equation (21) has $|M|$ linearly independent solutions, such as $(a_0/h_1, 0, \ldots, 0)$ etc. Therefore there exists a self facet (22) such that *the set of distinct values among a_j, $j \in N$ is $\{h_i, i \in M\}$*. The restriction of positivity of the h_i is not essential here. This example demonstrates the complexity of the convex hulls of general threshold sets, as compared with the well-described combinatorial polytopes such as matroids or matchings.

(4) The 0–1 master knapsack problem S_b is the set of 0–1 solutions $x = (x_i^j)$ of

$$\sum_{i=1}^{b} \sum_{j=1}^{K_i} i x_i^j \le b \quad (b = 1, 2, \ldots), \tag{27}$$

where $K_i = 1 + [b/i]$ is the smallest integer larger than b/i. Knowledge of conv(S_b) will provide the convex hull of every threshold set having a separator with a right-hand side of b [8]. Johnson [10] gave a procedure, based on sequential lift/extensions, to find many facets of conv(S_b). Hammer and Peled [9] computed all the facets of conv(S_b) for $b \le 7$. With S_b is associated the integer master problem T_b given by

$$\sum_{i=1}^{b} i t_i \le b, \quad t_i \ge 0, \text{ integer}. \tag{28}$$

Aráoz [1] studied the problem T_b and characterized its facets

$$\sum_{i=1}^{b} h_i t_i \le h_0. \tag{29}$$

The following result was pointed out by E. Johnson. for $h_1 > 0$, (29) is a facet of conv(T_b) if and only if the inequality

$$\sum_{i=1}^{b} \sum_{j=1}^{K_i} h_i x_i^j \le h_0 \tag{30}$$

is a facet of conv(S_b) ((30) is a facet with the special property that the coefficient of x_i^j does not depend on j; there are many other facets). To prove the result we use the linear transformation $t = Ax$, where $t_i = \sum_{j=1}^{K_i} x_i^j$. Clearly A maps S_b onto T_b, and (30) is valid for S_b if and only if (29) is valid for T_b. If (30) is a facet, then by the argument of Proposition 5 the hyperplane (29) is spanned by points $t = Ax$ such that x satisfies (30) with equality and (29) is a self facet. Conversely, assume that (29) is a self facet and let the rows of T be all points $t \in T_b$ that satisfy (29) with equality. If $n_i \ge 0$ is the largest entry in the i column of T, then $n_i \le [b/i]$ by (28), and therefore the n_i given by (26) satisfy $n_i \le K_i$. The argument of Proposition 6 shows that the hyperplane $\sum_{i=1}^{b} \sum_{j=1}^{n_i} h_i x_i^j = h_0$ is a self facet spanned by points that satisfy (27). Therefore (30), which is obtained from it by duplicating coefficients, is a facet of conv (S_1).

It is relatively simple to determine when a hyperplane (24) is spanned by integral points x. This happens if and only if (24) has some integral solution, or equivalently when the greatest common divisor of the b_i divides b_0. This was proven by Edelberg [5]. But if we add the restriction that $x_i \geq 0$, the problem is harder, and we do not know direct solutions to it.

Let us conclude this section with an example [11] of two different self facets that are separators of the same threshold set. It can be verified that the inequality

$$13x_1 - 7x_2 + 5x_3 + 6x_4 - 4x_5 + 4x_6 + 3x_7 - 2x_8 \leq 24$$

can be satisfied as an equality by 8 linearly independent 0–1 points. The solution set of this inequality is symmetric in x_7 and x_8. Therefore

$$13x_1 + 7x_2 + 5x_3 + 6x_4 - 4x_5 + 4x_6 - 2x_7 - 3x_8 \leq 24$$

is another self facet defining the same threshold set.

5. Chow parameters of self facets

We now look at the self facets from another point of view. Instead of asking, as in the previous section, what hyperplanes are facets of full-dimensional convex hulls of sets of binary points, we now ask what threshold sets S have a separator that is a facet of conv (S). It is convenient here to change the terminology slightly. First, we apply the transformation

$$y_j = 2x_j - 1, \quad j \in N, \tag{31}$$

which changes $\{0, 1\}$ into $\{-1, 1\}$, which we call now B. The new variables are more convenient for taking complements. Since (31) is an affine transformation, the image of a self facet is a hyperplane containing $|N|$ affinely independent points y, and conversely, so we may keep calling these images "self facets". Also, the image of a threshold set is the set of solutions y of a linear inequality, and conversely, so we may keep calling these images "threshold sets" and the corresponding inequalities "separators". Second, we represent each subset S of B^N by the Boolean function $F: B^N \to B$ having value -1 at the points of S and value 1 at the points of $B^N - S$. In particular, the Boolean functions representing threshold sets are called *threshold functions*.

Chow has discovered a set of parameters, now bearing his name, associated with every Boolean function F, such that if F is a threshold function, no other Boolean function with the same number of variables has the same parameters. We use here one variant of the Chow parameters, following Winder [17]. To define them we use the notion of dual Boolean functions.

Definition 5. If F is a Boolean function, its *dual* F^d is defined by $F^d(y) = -F(-y)$. F is *self dual* if $F(-y) = -F(y)$.

If $F : B^N \to B$ is any Boolean function and 0 is an index not in N, then the function $F^* : B^{0 \cup N} \to B$ given by

$$F^*(y_0, y) = y_0 \cdot F(y) + (-y_0) \cdot F^d(y)$$

is self dual. Here the multiplication operation $\cdot$ is that of taking the minimum and the addition operation $+$ is that of taking the maximum. It can be shown that the mapping $F \to F^*$ is a bijection of the Boolean functions on B^N onto the self dual Boolean functions on B^{N+1}, that $F = F^*$ if and only if F is self dual and that F^* retains many other properties of F, e.g. F^* is a threshold function if and only if F is a threshold function. For these reasons F^* may be thought of as a self dual version of F. We can now define the Chow parameters of F.

Definition 6. Let F be a Boolean function on B^N. If F is self dual, its *Chow parameter vector* p is given by

$$p_j = \frac{1}{4} \sum_{y \in B^N} y_j F(y), \quad j \in N. \tag{32}$$

If F is not self dual, its Chow parameter vector is that of F^*.

By using the self duality of F, we can rewrite (32) in the form

$$p = \frac{1}{4} \sum_{y} y F(y) = \frac{1}{4}\Big(\sum_{F(y)=1} y\Big) - \frac{1}{4}\Big(\sum_{F(y)=-1} y\Big) = \frac{1}{2} \sum_{F(y)=1} y,$$

so that p is proportional to the "center of gravity" of the points where $F = 1$. If G is another self dual function, obtained from F by complementing its values at a given point z and its complement $-z$, then the parameters of G are $q = p - zF(z)$. The parameter vector of the self dual function $F(y) = y_1$ is clearly $p = (2^{|N|-2}, 0, \ldots, 0)$, which is integral for $|N| \geq 2$ and a vector of even integers for $|N| \geq 3$. Since every other self dual function on B^N can be obtained from F by a sequence of changing the functional values at a pair of complementary points, it follows that all the components of a Chow parameter vector are integers for $|N| \geq 2$ and have the same parity for $|N| \geq 3$. Winder [17] shows that, when F is a self dual function with parameters p, a given q is a parameter vector of a self dual G if and only if there exists a set S of points y satisfying $F(y) = -1$ and $\sum_{y \in S} y = q - p$. This property can be used to characterize the Boolean functions F on B^N (whether self dual or not) such that no other Boolean function on B^N has the same parameters as F has. The characterizing property is that for no positive integer k do there exist *distinct* points $y^1, \ldots, y^k, z^1, \ldots, z^k$ of B^N satisfying $F(y^i) = 1$, $F(z^i) = -1$, $y^i \neq -z^j$ and $\sum_{i=1}^k y^i = \sum_{i=1}^k z^i$. In particular, then, threshold functions have this property, and so can be *labelled* by their Chow parameters. This was proved by Chow [4].

We may then rephrase the question at the beginning of this section as what parameter vectors label threshold functions having a self facet separator. We shall assume that the threshold function in question is self dual.

If F is a threshold function on B^N, it has a separator $a \cdot y = \sum_{i \in N} a_i y_i \leq b$ with integral a (since the requirements on a and b are homogeneous linear inequalities with integral coefficients). The dual function F^d has the separator $a \cdot y \leq -(b+1)$. If F is self dual, i.e. $F = F^d$, then the arithmetic mean of these two separators, namely $a \cdot y \leq -\frac{1}{2}$, is another separator of F. Since a is integral, $a \cdot y \leq -r$ is a separator of F for every $0 < r \leq 1$, and no point of B^N is orthogonal to a. Conversely, if no point of B^N is orthogonal to the integral a, then $a \cdot y \leq -r$ is a separator of the same self dual threshold function for all $0 < r \leq 1$. It is possible, but not easy, to find self dual threshold functions with a self facet separator of the form $a \cdot y \leq -r$ with $r \geq 2$ and a integral (of course, the greatest common divisor of the a_i is understood to be 1). This is translated into 0–1 variables to mean a self dual self facet $a \cdot x \leq b$ with $b \neq \frac{1}{2}(\sum_i a_i - 1)$. An example is given by Muroga [11] as $a = (29, 25, 19, 15, 12, 8, 8, 3, 3)$, $r = 2$. Since exactly six of the a_i are odd, $a \cdot y$ is even for all $y \in B^N$, so no y satisfies $a \cdot y = -1$. Nevertheless $a \cdot y = -2$ is a self facet.

In order to answer the question of this section, we denote by P^N the set of all parameter vectors of self dual Boolean functions on B^N, and $Q^N = \operatorname{conv}(P^N)$. Every vertex of Q^N is thus a parameter vector. The next theorem characterizes the faces of Q^N. Special cases of it, involving the vertices and facets of Q^N, are given by Winder [17].

Theorem 4. *Let a be a non-zero N-vector. Then the linear inequality $a \cdot w \leq b$ defines a d-dimensional face of Q^N ($d = 0, 1, \ldots, 2^{N-1} - 1$) if and only if*
(1) $b = a \cdot q$, where $q = \frac{1}{2}\sum_{a \cdot y > 0} y - \frac{1}{2}\sum_{a \cdot y < 0} y$ (b must be positive);
(2) the rank of the set $\{y \in B^N \mid a \cdot y = 0\} = \{\pm y^1, \ldots, \pm y^m\}$ is d.
In that case the set of parameter vectors on the face is

$$\left\{ q + \frac{1}{2}\sum_{i=1}^{m} u_i y^i : u \in B^m \right\}.$$

Proof. Clearly $a \cdot w \leq b$ defines a non empty face of Q^N (supports Q^N) if and only if

$$b = \max_{w \in Q^N} a \cdot w = \max_{w \in P^N} a \cdot w = \max_{G \text{ self dual}} \frac{1}{2} \sum_{y} (a \cdot y) G(y).$$

The self dual functions G that realize this maximum must be such that $G(y)$ and $a \cdot y$ have the same sign when $a \cdot y \neq 0$. In fact, for each $u \in B^m$, let F^u be the self dual Boolean function defined by $F^u(y) = 1$ if $a \cdot y > 0$, $F^u(y) = -1$ if $a \cdot y < 0$ and $F^u(y^i) = u_i$, $F^u(-y^i) = -u_i$. Then these F^u are all the self dual functions G that realize the above maximum. It follows that $b = a \cdot q$, so that the face is non empty if and only if (1) holds. (That $a \cdot q$ is positive follows from the fact that if

a is orthogonal to all y, then $c = 0$.) Moreover, the parameter vectors on the face are precisely the parameters of the F^u, which by (52) are given by

$$q - \tfrac{1}{2}\sum_{i=1}^{m} y^i u_i + \tfrac{1}{2}\sum_{i=1}^{m} (-y^i)(1 - u_i) = q - \tfrac{1}{2}\sum_{i=1}^{m} u_i y^i .$$

To show (2), please note that since the face is spanned by $\{q - \frac{1}{2}\sum_{i=1}^{m} u_i y^i \mid u \in B^m\}$, $d - 1$ is equal to the affine rank of $\{\sum_{i=1}^{m} u_i y^i \mid u \in B^m\}$ and d is equal to its linear rank (since the y^i are orthogonal to a). Let the rows of Y be $y^1, \ldots, y^m$ and the rows of U the 2^m m-vectors u. Then d is the rank of UY. Since the rank of U is m, there exists a non singular 2^m by 2^m matrix R such that $U = R\binom{I}{0}$, where I is the m by m identity matrix. Hence

$$\operatorname{rank}(UY) = \operatorname{rank}(\tbinom{I}{0} Y) = \operatorname{rank}(Y) = \operatorname{rank}(Y). \quad \square$$

Corollary 5. *The vertices of Q^N are precisely the parameter vectors p of the self dual threshold functions F on B^N. An inequality $a \cdot y \leq 0$ is a separator of F if and only if $a \cdot w \leq a \cdot p$ supports Q^N only at $w = p$.*

Proof. If $d = 0$ then $m = 0$, so that a is not orthogonal to any $y \in B^N$. Therefore $a \cdot y \leq 0$ is a separator of a self dual threshold function F and the parameter vector of F is just q, which is also the unique point on the face. Conversely, let F be a self dual threshold function. Then F has a separator of the form $a \cdot y \leq 0$. Any such a is not orthogonal to any $y \in B^N$, and hence $m = 0$ and $a \cdot w \leq a \cdot q$ is a 0-dimensional face of Q^N, i.e. a vertex. The vertex is q, which is also the parameter vector of F. $\square$

Corollary 6. *An edge of Q^N contains exactly two parameter vectors, namely its extreme points. The difference between them is a vector $y^1 \in B^N$, and the self dual threshold functions labelled by the two parameter vectors differ only at $\pm y^1$.*

Proof. Let $a \cdot w \leq b$ be an edge of Q^N. Then the rank of Y, the matrix of non-complementary vectors of B^N orthogonal to a, is 1, so that Y has only one row y^1. The edge then contains at most two distinct parameter vectors $q + \frac{1}{2}y^1$ and $q - \frac{1}{2}y^1$, hence it contains exactly these parameter vectors. The self dual Boolean functions whose parameter vectors lie on the edge are the two functions G^1 and G^{-1}, which differ only at $\pm y^1$. The parameter vectors are vertices of Q^N, hence G^1 and G^{-1} are threshold functions and are uniquely determined by their parameters. $\square$

Theorem 5. *Let $a \cdot y \leq -r < 0$ be a separator of the self dual threshold function labelled by p. If $q \in Q^N$, $q - p = y^* \in B^N$ and $a \cdot y^* = -r$, then q is a vertex of Q^N adjacent to p and for every t ($n < t \leq r/(n-2)$ (where $n = |N| \geq 3$), $(a + ty^*) \cdot y \leq 0$ is a separator of the self dual threshold function labelled by q.*

Proof. Let p label the self dual threshold function F. Let $p^1, \dots, p^k$ be all the vertices of Q^N adjacent to p and let them label the self dual threshold functions $F^1, \dots, F^k$, respectively. By Corollary 6 each F^i differs from F only at a pair of complementary points $y^i, -y^i$, and by choosing y^i so that $F(y^i) = -1$ we have $p^i = p - y^i F(y^i) = p + y^i$. Therefore $a \cdot p^i \geq a \cdot p - r$. Since $q \in Q^N$, $y^* = q - p$ is a non-negative combination of the extreme directions $y^i = p^i - p$, i.e. there exist non-negative numbers $c_1, \dots, c_k$ such that $y^* = \sum_1^k c_i y^i$. If $\sum_1^k c_i > 1$, then $a \cdot y^* \leq -r \sum_1^k c_i < -r$, contradicting our assumption. Therefore $\sum_1^k c_i \leq 1$. This means that the ± 1 vector y^* is a convex combination of the ± 1 vectors $y^1, \dots, y^k$ and the origin. But the ± 1 vectors are the extreme points of the hypercube $|y_i| \leq 1$ and the origin is an internal point. It follows that one of the c_i must be 1 and y^* is one of the y^i, say y^1, so that $q = p^1$. To prove that $(a + ty^*) \cdot y \geq 0$ is a separator of F^1 we use the fact that F^1 differs from F only at $\pm y^1$. At the point $y = y^*$ one has $(a + ty^*) \cdot y = (a + ty^*) \cdot y^* = -r + tn > 0$. At any other point $y \neq y^*$ such that $F(y) = -1$ one has $a \cdot y \leq -r$ and therefore $(a + ty^*) \cdot y \leq -r + t(n - 2) < 0$. At the complementary points $-y$ we have of course the opposite inequalities. Thus the self dual threshold function with the separator $(a + ty^*) \cdot y \geq 0$ is identical with F^1.

We are now ready to characterize those vertices of Q^N that label self dual threshold functions with self facet separators.

Theorem 6. *Let F be a self dual threshold function on B^N with Chow parameter vector p. If F has a self facet separator of the form*

$$a \cdot y \geq -r, \tag{33}$$

then there exist $|N|$ affinely independent vertices q^i of Q^N adjacent to p such that

$$a \cdot q^i = a \cdot p - r \tag{34}$$

Conversely, if F has a separator (33) with $r > 0$, and if (34) is satisfied by N affinely independent Chow parameter vectors q^i, then (33) is a self facet and the q^i are vertices of Q^N adjacent to p.

Proof. Assume that (33) is a self facet separator of F. Then $r > 0$ and there exist $|N|$ linearly independent vectors $y^i \in B^N$ satisfying (33) with equality. The self dual Boolean function that differs from F only at $\pm y^i$ has a parameter vector $q^i = p - y^i F(y^i) = p + y^i$. These q^i satisfy (34). By Theorem 5 they are vertices of Q^N adjacent to p. They are affinely independent because the y^i are (if F is monotone, then $p > 0$ and the q^i are in fact linearly independent).

Conversely, assume that $r > 0$, (33) is a separator of F and there exist $|N|$ affinely independent parameter vectors q^i satisfying (34). Then by an earlier remark, for each i there exists a non-empty set $S^i \subset B^N$ of points y such that $F(y) = -1$ and $\sum_{y \in S^i} y = q^i - p$. By (33) and (34) we then have

$$r^i(S^i) = \sum_{j \in S^i} a_j y_j - a \cdot (q^i - p) = r.$$

This shows that S^i is a singleton $\{y^i\}$, and $q^i = p + y^i$. By (34) the y^i satisfy (33) with equality and they are affinely independent because the q^i are. By Theorem 5 the q^i are vertices of Q^N adjacent to p.

Example. Consider the self dual threshold function having the self facet separator $a \cdot y \gtrless -1$ with $a = (3, 3, 2, 2, 2, 1)$. Its Chow parameter vector is $p = (7, 7, 5, 5, 5, 1)$ and $a \cdot p = 73$. The points $y \in B^6$ satisfying $a \cdot y = -1$ are

$(1, 1, -1, -1, -1, -1)$ (1 point),
$(1, -1, 1, -1, -1, 1)$ etc. (6 points),
$(-1, -1, 1, 1, 1, -1)$ (1 point).

These 8 points are the rows of a matrix with rank 6. Adding each of them to p, we get the following parameter vectors:

$(8, 8, 4, 4, 4, 0)$ (1 vector),
$(8, 6, 6, 4, 4, 2)$ etc. (6 vectors),
$(6, 6, 6, 6, 6, 0)$ (1 vector).

These 8 vectors q satisfy $a \cdot q = 72$ and are the rows of a matrix with rank 6. By Theorem 5 they are vertices of Q^6 adjacent to p, hence parameter vectors of self dual threshold functions. Indeed they label the functions with separators $b \cdot y \gtrless -1$, where b is, respectively,

$(2, 2, 1, 1, 1, 0)$ (1 function),
$(4, 2, 3, 2, 2, 1)$ etc. (6 functions),
$(1, 1, 1, 1, 1, 0)$ (1 function).

Acknowledgements

I am grateful to Professors Peter Hammer, Ellis Johnson and Jack Edmonds for stimulating and illuminating discussions on the problems of this paper. The referees made valuable suggestions for improving an earlier version [14]. This work was supported by NRC Postgraduate Scholarship.

References

[1] J.A. Araoz Durand, Polyhedral neopolarities, Ph.D. Thesis, University of Waterloo, Faculty of Mathematics, Department of Applied Analysis and Computer Science, November 1974.
[2] E. Balas, Facets of the knapsack polytope, *Math. Programming* 8 (1975) 146–164.
[3] E. Balas and E. Zemel, Facets of the knapsack polytope from minimal covers, *Carnegie-Mellon University, Management Science Research Report*, No. 352, December 1974.

[4] C.K. Chow, On the characterization of threshold functions, in *Switching Circuit Theory and Logical Design*, IEEE Special Publication S-134, September 1961, pp. 34–38.

[5] M. Edelberg, Integral convex polyhedra and an approach to integralization, Ph.D. Thesis, *Massachusetts Institute of Technology, Electrical Engineering Department*, August 1970.

[6] C.C. Elgot, Truth functions realizable by single threshold organs, in *Switching Circuit Theory and Logical Design*, IEEE Special Publication S-134, September 1961, pp. 225–245.

[7] P.L. Hammer, E.L. Johnson and U.N. Peled, Facets of regular 0–1 polytopes, *Math. Programming* 8 (1975) 179–206.

[8] P.L. Hammer, E.L. Johnson and U.N. Peled, The role of master polytopes in the unit cube, *University of Waterloo, Combinatorics and Optimization Research Report*, CORR 74–25, October 1974, forthcoming in *SIAM J. Appl. Math.*

[9] P.L. Hammer and U.N. Peled, Computing low capacity 0–1 knapsack polytopes, *University of Waterloo, Combinatorics and Optimization Research Report*, CORR 75-5, February 1975.

[10] E.L. Johnson, A class of facets of the master 0–1 knapsack polytope, *IBM Thomas J. Watson Research Center*, RC 5106, 1974.

[11] S. Muroga, *Threshold logic and its applications* (Wiley, New York, 1971).

[12] G.L. Nemhauser and L.E. Trotter, Jr., Properties of vertex packing and independence system polyhedra, *Math. Programming* 6 (1974) 48–61.

[13] M.W. Padberg, On the facial structure of set packing polyhedra, *Math. Programming* 5 (1973) 199–215.

[14] U.N. Peled, Unfixed variables, self facets and Chow parameters, *University of Waterloo, Combinatorics and Optimization Research Report*, CORR 75–16, July 1975.

[15] M.A. Pollatschek, Algorithms on finite weighted graphs, Ph.D. Thesis, *Technion — Israel Institute of Technology, Faculty of Industrial and Management Engineering*, 1970 (in Hebrew, with English Synopsis).

[16] L.E. Trotter, Jr., A class of facet producing graphs for vertex packing polyhedra, *Discrete Math.* 12 (1975) 373–388.

[17] R.O. Winder, Chow parameters in threshold logic, *J.A.C.M.* 18 (1971) 265–290.

[18] R.O. Winder, Threshold logic, Ph.D. Thesis, *Princeton University, Department of Mathematics*, 1962, published on demand by *University Microfilms, Xerox Company*, Ann Arbor, Mich., 1971.

[19] L.A. Wolsey, Faces for a linear inequality in 0–1 variables, *Math. Programming* 8 (1975) 165–178.

[20] L.A. Wolsey, Facets and strong valid inequalities for integer programs, CORE, *University of Louvain*, April 1974.

[21] E. Zemel, Lifting the facets of 0–1 polytopes, *Carnegie Mellon University, Management Science Research Report*, No. 354, December 1974, Revised December 1975.

Annals of Discrete Mathematics 1 (1977) 457-464
© North-Holland Publishing Company

VERTEX GENERATION METHODS FOR PROBLEMS WITH LOGICAL CONSTRAINTS

David S. RUBIN

Graduate School of Business Administration, University of North Carolina, Chapel Hill, NC 27514, U.S.A.

Recent work has shown how to use vertex generation methods to solve linear complementarity problems and cardinality constrained linear programs. These problems can be characterized as linear programs with additional logical constraints. These logical constraints can be incorporated into Chernikova's vertex generating algorithm in a natural and straightforward fashion. This study examines the extension of this technique to other linear programs with logical constraints, and discusses its use as a solution procedure for the 0-1 integer programming problem.

We wish to consider the convex polyhedral set $F = \{x \mid Ax \leq b, x \geq 0\}$, where A and b are given real matrices of order $m \times n$ and $m \times 1$, respectively, and x is an $n \times 1$ vector of real variables. Introducing slack variables s, we may imbed F into a higher dimensional space as

$$F' = \left\{ y = \binom{s}{x} \,\Big|\, Ax + s = b, x \geq 0, s \geq 0 \right\}.$$

We shall be interested in the sets of vertices of F and F'. Since a natural and obvious correspondence exists between the vertices of these two polyhedra (and indeed between all of the points in these polyhedra) we shall henceforth refer to both of these sets as F.

There are many problems in mathematical programming whose feasible region is a polyhedron F, and whose optimal solution is a vertex of F. In theory, any such problem can be solved by determining all the vertices of F and then choosing the best of this finite set. However, since the number of vertices of F grows exponentially in m and n, such a procedure is not practical except for small problems.

Problems such as the cardinality constrained linear program [10,12] (see also [3,7] where it is called the generalized lattice point problem) and the linear complementarity problem have optimal solutions which are vertices of F and which satisfy additional conditions which we refer to as "logical constraints."

Let L_i be a subset of $\{1, 2, \ldots, m+n\}$ for $i = 1, \ldots, k$. Associated with each L_i is an integer $q_i < |L_i|$. Let y be any point of F and let y^i denote that subvector of y containing those components of y with indices in L_i. Let $|w|^+$ denote the number of positive components of an arbitrary vector w. Then the logical constraints are

$$|y^+ L_i| \le q_i, \quad i = 1, 2, \ldots, k.$$

(We do not assume that the sets L_i are disjoint, nor that they exhaust $\{1, 2, \ldots, m+n\}$. These assumptions hold in some problems of interest, e.g., the linear complementarity problem, but the procedure we shall present is valid whether they hold or not.) It is easy to show that if a linear program with logical constraints is feasible and bounded, then at least one vertex of F will be optimal.

The body of this paper shows how to modify Chernikova's vertex generating algorithm [4, 5] to generate only that subset of the vertices of F which also satisfy the logical constraints. To the extent that this is a small subset, the procedure will be practical; if the subset is large, it will not be useful. In the cardinality constrained linear program, there is only one logical constraint, with $L_1 = \{m+1, \ldots, m+n\}$. If $q_1 = 1$, there are at most $2n$ vertices satisfying the logical constraint; but if $q_1 \ge \min\{m, n\}$, then all vertices of F satisfy the logical constraint. In general, the strength of the logical constraints (in terms of the number of vertices of F which they exclude) in particular problems is a topic that, to the best of our knowledge, has not been studied.

Rather than concentrating on the logically feasible vertices of F, it is possible to approach these problems by studying the convex hull of the feasible points of a linear program with logical constraints. In reference [2], Balas has given a characterization of the convex hull. Other discussions of linear programs with logical constraints can be found in references [1, 3, 6–8, 10–12].

Section 1 presents Chernikova's algorithm. Since this material is available elsewhere, it is included here only to make the present paper self-contained. Section 2 shows how to modify that algorithm to incorporate the logical constraints; it is an extension and generalization of work found in references [9, 10, 11]. Section 2 also shows how to incorporate the objective function of the problem, if one exists, so that one generates only vertices better than those previously generated.

In Section 3 we discuss the geometry of the procedure and contrast our work with the cutting-plane methods of Balas [1, 2] and Glover et al. [7, 8]. This leads to Section 4, which investigates the application of the technique to the 0-1 integer program. Finally, in Section 5 we briefly discuss further modification of the algorithm to incorporate logical constraints of the form $|y^+ L| = q$ and $|y^+ L| \ge q$.

1. Chernikova's algorithm

Chernikova has given an algorithm [4, 5] which calculates all the edges of a convex polyhedral cone in the nonnegative orthant with vertex at the origin. This algorithm can also be used to find all the vertices of F by virtue of the following easily proved lemma:

Lemma 1. *x is a vertex of $F = \{x : Ax \le b, x \ge 0\}$ if and only if*

$$\left\{ \begin{pmatrix} \lambda x \\ \lambda \end{pmatrix}, \lambda \geq 0 \right\}$$

is an edge of the cone

$$C_1 = \left\{ \begin{pmatrix} x \\ \xi \end{pmatrix} \,\middle|\, (-A, b)\begin{pmatrix} x \\ \xi \end{pmatrix} \geq 0, \begin{pmatrix} x \\ \xi \end{pmatrix} \geq 0 \right\}.$$

Here ξ and λ are scalar variables.

We shall accordingly concern ourselves with finding all the edges of sets of the form $C = \{w : Dw \geq 0, w \geq 0\}$, where D is $p \times q$.

Consider the matrix $\binom{D}{I}$ where I is a $q \times q$ identity matrix. Chernikova's algorithm gives a series of transformations of this matrix which generates all the edges. At any stage of the process we denote the old matrix by $Y = \binom{U}{L}$, and the new matrix being generated is denoted $\bar{Y}$. The matrices U and L will always have p and q rows, respectively; however, they will in general not have q columns. They will have more than q columns in most cases, but if C lies in some subspace of $\mathbf{R}^q$ they may have fewer than q columns. For $w \in \mathbf{R}^q$, we use the symbol (w) to denote the ray $\{\lambda w, \lambda \geq 0\}$.

The algorithm is as follows:

0.0. If any row of U has all components negative, then $w = 0$ is the only point in C.

0.1. If all the elements of U are nonnegative, then the columns of L are the edges of C, i.e., the ray (l_j) is an edge of C; here l_j denotes the j^{th} column of L.

1. Choose the first row of U, say row r, with at least one negative element.

2. Let $R = \{j : y_{rj} \geq 0\}$. Let $v = |R|$, i.e., the number of elements of R. Then the first v columns of the new matrix, $\bar{Y}$, are all the y_j for $j \in R$, where y_j denotes the j^{th} column of Y.

2′. If Y has only two columns and $y_{r1} \cdot y_{r2} < 0$, adjoin the column $|y_{r2}| y_1 + |y_{r1}| y_2$ to the $\bar{Y}$ matrix. Go to step 4.

3. Let $S = \{(s, t) : y_{rs} y_{rt} < 0, s < t\}$, i.e., the set of all (unordered) pairs of columns of Y whose elements in row r have opposite signs. Let I_0 be the index set of all nonnegative rows of Y. For each $(s, t) \in S$ find all $i \in I_0$ such that $y_{is} = y_{it} = 0$. Call this set $I_1(s, t)$. We now use some of the elements of S to create additional columns for $\bar{Y}$:

(a) If $I_1(s, t) = \emptyset$ (the empty set), then y_s and y_t do not contribute another column to the new matrix.

(b) If $I_1(s, t) \neq \emptyset$, check to see if there is a u not equal to either s or t such that $y_{iu} = 0$ for all $i \in I_1(s, t)$. If such a u exists, then y_s and y_t do not contribute another column to the new matrix. If no such u exists, then choose $\alpha_1, \alpha_2 > 0$ to satisfy $\alpha_1 y_{rs} + \alpha_2 y_{rt} = 0$. (One such choice is $\alpha_1 = |y_{rt}|$, $\alpha_2 = |y_{rs}|$.) Adjoin the column $\alpha_1 y_s + \alpha_2 y_t$ to the new matrix.

4. When all pairs in S have been examined, and the additional columns (if any) have been added, we say that row r has been "processed." Now let Y denote the matrix $\bar{Y}$ produced in processing row r, and return to step 0.0.

The following remarks about the algorithm will be useful later.

(1) Let C_i be the cone defined by $C_i = \{w \mid D^i w \geq 0, w \geq 0\}$, where D^i is composed of the first i rows of D. Let $C_0 = \{w \mid w \geq 0\}$ and $C_p = C$. Then $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_p$, and each cone differs from its predecessor in having one additional defining constraint. The algorithm computes the edges of $C_0, C_1, \ldots, C_p$ successively by adding on these additional defining constraints. Clearly the edges of C_0 are the unit vectors. After the algorithm has processed row i, the L matrix has all the edges of C_i as its columns.

(2) Let d^i denote the i^{th} row of D. Then initially $u_{ij} = d^i l_j$ and by linearity this property is maintained throughout the algorithm. Thus u_{ij} is the slack in the constraint $d^i l_j \geq 0$. In particular, if $d^i = (-a^i, b_i)$ and $l_j = \binom{x}{\xi}$, then u_{ij} is the slack in the constraint $a^i x \leq b_i$, i.e., in the ith constraint of $Ax \leq b$, when $\xi = 1$.

2. Modifications of Chernikova's algorithm

From Lemma 1, we see that we want only those edges of C_p that have
Since the defining inequalities of C_p are homogeneous, the edges constructed by the algorithm can be normalized after the algorithm terminates. We prefer, however, to do the normalization as the algorithm proceeds. Accordingly, whenever an edge is created with $\xi > 0$, it will be normalized to change the ξ value to one.

When applying Chernikova's algorithm to find the edges of C_p, let $y_j = \binom{u_j}{l_j}$ be the j^{th} column of Y. Let y_j^+ be that subvector of y_j containing those components of y_j whose indices are in the set I_p. Finally, let $y_j^+(r)$ be that subvector of y_j^+ whose indices are in the set $\{1, 2, \ldots, r-1; m+1, m+2, \ldots, m+n\}$.

Lemma 2. *Suppose that in processing row r we produce a column y_s with $|y_s^+(r)|^+ \geq q_p$. Then any column y_t subsequently produced as a linear combination of y_s and some other y_k will also have $|y_t^+(r)|^+ \geq q_p$.*

Proof. The algorithm creates new columns by taking strictly positive linear combinations of two old columns. Since $L \geq 0$ and the first $r-1$ rows of U are nonnegative after row $r-1$ has been processed, the new $y_t^+(r)$ will have at least as many positive components as the old $y_s^+(r)$. ∎

Lemma 3. *Suppose that in processing row r we encounter the following situation: $y_{rs} < 0$, $y_{rt} > 0$ and there exist k and l such that $y_{ik} = 0$ for all $i \in I_1(s,t)$ and $|y_k^+(r)|^+ \geq q_p$. For any $\alpha_1 > 0$ and $\alpha_2 > 0$, let $y_v = \alpha_1 y_s + \alpha_2 y_t$. Then $|y_v^+(r)|^+ \geq q_p$.*

Proof. Suppose y_{uk} is a strictly positive component of $y_k^+(r)$. Since $y_{ik} = 0$ for all

$i \in I_1(s, t)$, it follows that $u \notin I_1(s, t)$. Hence at least one of y_{iu}, y_{iv} is strictly positive, and since $\alpha_1, \alpha_2 > 0$, we have $y_{iw} > 0$. Thus $|y_w^i(r)|^+ \geq |y_u^i(r)|^+ > q_i$. $\square$

Theorem 1. *If while processing row r the algorithm ever produces a column with any $|y_k^i(r)|^+ > q_i$, that column may be discarded from further computation.*

Proof. The theorem follows immediately from Lemmas 2 and 3 by induction. $\square$

If we actually had to enumerate all the edges of C_P it would be impractical to use the Chernikova algorithm as a procedure to find the vertices of P satisfying the logical constraints. To repeat what was said earlier, however, to the extent that the logical conditions eliminate many of the vertices of C_P, Theorem 1 will permit considerable savings of storage and time. Consider the linear complementarity problem (LCP)

$$Ax + s = b$$
$$x, s \geq 0$$
$$x^T s = 0.$$

Here A is $m \times m$, and there are m logical constraints with $L_i = \{i, m+i\}$ and $q_i = 1$. If $A = I$, the identity matrix, and $b > 0$, then P has 2^m vertices, all of which satisfy the logical constraints. On the other hand, any strictly convex quadratic program gives rise to an LCP whose logical constraints are so strong that only a single vertex of P satisfies them.

In the LCP, we are interested only in finding some vertex which satisfies the logical constraints. However, in other problems such as the cardinality constrained linear program, there is a linear objective function $c^T x$ which is to be maximized. By introducing the objective function into consideration, we can try to achieve savings besides those indicated by Theorem 1.

Lemma 4. *Suppose that we have processed row r and that $y_{ri} > 0$, $y_{r+1,i} = 1$. Then x_i is a vertex of P.*

Proof. We know l_i is an edge of C_r. Since $u_i > 0$, l_i satisfies $-Ax + b\xi \geq 0$, so $l_i \in C_P$. Since l_i is an edge of C_r and $C_P \subset C_r$, l_i is also an edge of C_P. It now follows from Lemma 1 that x_i is a vertex of P. $\square$

Suppose that after processing row r we have found a vertex of C_P with $c^T x = \beta$. We could now add the constraint $c^T x \geq \beta$ to the constraints $Ax \leq b$. This is a simple matter to do: We can initially include the vector $(c^T\ 0)$ as the zero$^{\text{th}}$ row of U. Thus y_{0i} will be the value of $c^T x_i$. When we find a vertex with $c^T x = \beta$, we modify the zero$^{\text{th}}$ row to represent the constraint $c^T x \geq \beta$. To do that we need only change y_{0j} to $y_{0j} - \beta y_{m+k+1,j}$ and now we treat the zero$^{\text{th}}$ row as another constraint and can apply the algorithm to it as well.

Subsequently we may produce a column with $y_{0k} \geq 0$, $u_k \geq 0$, $y_{m+k+1,k} = 1$. Hence we have found a vertex with $c \cdot x = \beta + y_{0k} \geq \beta$. We can now change all y_{0j} to $y_{0j} - y_{0k} y_{m+k+1,j}$ and again treat the zero$^{\text{th}}$ row as a constraint. Continuing in this fashion we will only generate vertices at least as good as the best vertex yet found. If we let γ be the sum of the amounts which we have subtracted from the y_{0j}, then we can recover the true optimal value of the objective by adding γ to the final value of y_{0j} in the column representing the optimal vertex.

It is not at all clear that using the objective function in this manner will make the procedure more efficient. Introducing the objective as a cutting plane in this fashion does exclude some vertices of F from consideration, but it may also create new vertices. It is impossible to tell a priori whether there will be a net increase or decrease in the number of vertices.

3. The geometry of logical constraints

The polyhedron $F = \{y = \binom{x}{s} \mid Ax + s = b, y \geq 0\}$ lies in the nonnegative orthant in $\mathbf{R}^{m+n}$. Each logical constraint says that of the variables in the set L_i at most q_i can be strictly positive, or alternatively, at least $|L_i| - q_i$ of these variables must be equal to 0. Thus each logical constraint excludes all vertices of F except those lying on a subset of the faces of the nonnegative orthant in $\mathbf{R}^{m+n}$. Since each constraint $y_i \geq 0$ defines a facet of the nonnegative orthant, and since the hyperplane $\{y : y_i = 0\}$ either supports F or else has no intersection with F, it follows that the logical constraints restrict the feasible region of the problem to a union of some of the faces of F. Thus the feasible region is a union of convex polyhedra that in general is not itself convex.

The test given in Theorem 3 determines whether a column to be generated does lie on one of the permitted faces of the orthant. In effect the modified Chernikova algorithm is simultaneously finding all the vertices of a collection of convex polyhedra and automatically excluding from consideration those vertices of F which do not lie on the "logically feasible faces." The structure of the set of logically feasible faces for the 0–1 integer program is discussed further in the next section.

The work of Balas [1, 2] and Glover et al. [7, 8] discusses classes of problems which include our linear programs with logical constraints. Using the objective function of the problem, they find the best vertex of F. If that vertex does not satisfy the logical conditions, they add an intersection cut (also called a convexity cut) derived from the constraints defining F and the logical constraints. This constraint is valid for all the logically feasible faces of F. Thus their procedures work with all of F and then cut away regions in F that are not logically feasible. These procedures can be characterized as dual algorithms. In contrast, our procedure considers only logically feasible vertices of F and can be characterized as a primal algorithm.

4. The zero-one integer program

We consider the problem

$$\begin{aligned} &\max\ c^T x \\ &\text{subject to}\ Dx \leq d \\ &\qquad\qquad Ix \leq e \\ &\qquad\qquad x \text{ integer}, \end{aligned}$$

where D is a real $(m-n)\times n$ matrix, d is a real $(m-n)\times 1$ vector, I is the $n\times n$ identity matrix and e is a vector of n ones. Introducing slack variables s and t to the constraints $Dx \leq d$ and $Ix \leq e$, respectively, our integer program can be viewed as a linear program with logical constraints:

$$F = \left\{ y = \begin{pmatrix} x \\ s \\ t \end{pmatrix}^T : Dx + s = d,\ Ix + t = e,\ y \geq 0 \right\},$$

$$L_i = \{m-n+i, m+i\}, \quad q_i = 1 \quad \text{for } i = 1, 2, \ldots, n.$$

The initial tableau for the algorithm is

$$\left(\frac{U}{L}\right) = \begin{pmatrix} -D & d \\ -I & e \\ \cdots & \\ I & 0 \\ 0 & 1 \end{pmatrix}.$$

Lemma 5. *At all stages of the process $u_{m-n+k,j} = l_{n+1,j} - l_{kj}$ in each column j, for all $k = 1, 2, \ldots, n$.*

Proof. Clearly the condition holds in the initial tableau. It follows by linearity and induction that it holds for all columns subsequently produced. $\square$

The import of the lemma is that there is no need to carry along those rows of L corresponding to the initial identity matrix. They can always be reconstructed from the last n rows of U and the final row of L.

Lemma 6. *We may assume without loss of generality*

(a) d^1, *the first row of* D, *is strictly positive.*

(b) d_1, *the first component of* d, *is strictly positive.*

(c) *for each component* d_{1j} *of the first row of* D *we have* $d_{1j} \leq d_1$.

Proof. By interchanging the names of the pair $(x_j, \bar{x}_j)$, if necessary, we can guarantee that the first nonzero component of each column of D is strictly positive. (If any column of D contains all zero entries, we may eliminate that variable from the problem.) By taking appropriate nonnegative weights for the rows of D, we can create a surrogate constraint with strictly positive coefficients. Listing this constraint first gives us part (a). If $d_1 \leq 0$, then F is empty or else $F = \{0\}$. In either case the problem is uninteresting, which proves (b). If $d_{1j} > d_1$, then $x_j = 0$ in any feasible solution, and so it may be eliminated from the problem, proving (c). □

Let us initiate the algorithm by processing row 1. Thus column $n+1$ is retained, and each column y_j for $j = 1, \dots, n$ is replaced by

$$\frac{d_1}{d_{1j}} y_j + \begin{pmatrix} \bar{d} \\ \varepsilon \\ 0 \\ 1 \end{pmatrix}.$$

In particular we now have $u_{n+1,j} = -1$ for all j and hence by Lemma 5, $u_{k,j} + u_{n+k,j} + 1 = 0$, for each column j and all $k = 1, \dots, n$. Furthermore, it follows from part (c) of Lemma 6 that each entry in the last n rows of U either is negative or else is equal to $+1$. (In fact the only negative entries are $u_{n+1+j,j}$ for $j = 1, 2, \dots, n$, but we shall not use this fact.) The remark in the first paragraph of Section 2 now tells us that all subsequent columns produced will be convex combinations of two other columns, and so it follows by induction that

(1) All entries in row $n+1$ of L will always be -1, and hence we may discard the entire L matrix.

(2) All entries in the last n rows of U will always be at most $+1$.

In the statement of Chernikova's algorithm and its modifications, it was convenient to assume that the rows of A were processed sequentially from the top down. However, it is clear that they can be processed in any order. The amount of work needed on any given problem can vary greatly, depending on the order in which the rows are processed, but there seems to be no a priori way to determine an efficient order. A myopic heuristic is given in [10]. Since the logical constraints in the 0–1 integer program involve the x and $\bar{x}$ variables, we cannot use the logical constraints to eliminate columns until we process some of the last n rows of U. Then after we have processed any of those rows, Theorem 1 can be rephrased as

Theorem 2. *After row $m-n+k$ of U has been processed, all columns with $0 < u_{m-n+k,j} < 1$ can be discarded.*

The remaining columns can be divided into two sets, those with $u_{m-n+k,j} = 0$ and those with $u_{m-n+k,j} = 1$. Theorem 2 now tells us that no column in one of these sets will ever be combined with any column in the other set. This is perhaps best understood in terms of the logically feasible faces discussed in Section 3. Each logical constraint in this problem defines a set of two logically feasible faces which are parallel to each other, and hence no convex combination of two points, one on each face, can itself be a feasible point for the problem. This result is not specific to the 0–1 integer program, but will hold in any problem whose logical constraints give rise to a set of disjoint logically feasible faces such that each feasible vertex must lie on at least one of the faces in the set.

Once row $m-n+k$ has been processed, there are now two polyhedra of interest

$$F_1 = F \cap \{y : x_k = 1\}, \qquad F_0 = F \cap \{y : x_k = 0\}.$$

Furthermore, we may, if we wish, work exclusively on F_1 or F_0, thereby reducing the active storage required to implement the procedure. Then the only information about F_1 that will be used in working on F_0 will be information about the objective function as discussed in Lemma 4 and the subsequent comments. It should also be remarked that the splitting of F into F_0 and F_1 (and an irrelevant part between F_0 and F_1) and the subsequent separate processing of F_0 and F_1 will result in an algorithm that is similar in spirit to standard implicit enumeration algorithms.

5. Other logical constraints

We will conclude with a few brief remarks about extending the results of Section 2 to logical constraints of the forms $|y^{L_i}|^+ = q_i$ and $|y^{L_i}|^+ \le q_i$. First of all we note that such constraints may give rise to problems which fail to have optimal solutions even though they are feasible and bounded. Consider the example

$$\begin{aligned} \max\quad & y_1 + y_2 \\ \text{subject to}\quad & y_1 + y_3 = 1 \\ & y_2 + y_4 = 1 \\ & y \ge 0 \\ & L_1 = \{3,4\},\ q_1 = 1 \end{aligned}$$

If the logical constraint is $|y^{L_1}|^+ = 1$, then feasible points with objective value arbitrarily close to 2 lie on the segments $y_1 = 1$ and $y_2 = 1$, but the point $(1,1,0,0)$ is infeasible. A similar result holds if the logical constraint is $|y^{L_1}|^+ \le 1$. Clearly vertex generation methods will be useless for such problems.

Let us then consider the more restricted problem of finding the best vertex of P subject to these new logical constraints. Clearly Lemmas 2 and 3 and Theorem 1 apply as stated for constraints $|y^+| = q_i$. However, since columns with $|y^+| \geq q_i$ can be constructed from columns with $|y^+| < q_i$ it does not appear that Theorem 1 can be strengthened for constraints $|y^+| = q_i$. Similarly we can see that there are no results analogous to Theorem 2 for constraints $|y^+| \geq q_i$. For such constraints, the best we can do is to use Chernikova's algorithm to generate all the vertices of P, and this is admittedly not particularly efficient.

References

[1] E. Balas, Intersection cuts from disjunctive constraints, Management Sciences Research Report No. 330, Carnegie-Mellon University, February 1974.

[2] E. Balas, Disjunctive programming: Properties of the convex hull of feasible points, Management Sciences Research Report No. 348, Carnegie-Mellon University, July 1974.

[3] A.V. Cabot, On the generalized lattice point problem and nonlinear programming, *Operations Res.*, 23 (1975) 565–571.

[4] N.V. Chernikova, Algorithm for finding a general formula for the nonnegative solutions of a system of linear equations, *U.S.S.R. Computational Mathematics and Mathematical Physics*, 4 (1964) 151–158.

[5] N.V. Chernikova, Algorithm for finding a general formula for the nonnegative solutions of a system of linear inequalities, *U.S.S.R. Computational Math. and Math. Phys.*, 5 (1965) 228–233.

[6] C.B. Garcia, On the relationship of the lattice point problem, the complementarity problem, and the set representation problem, Technical Report No. 145, Department of Mathematical Sciences, Clemson University, August 1973.

[7] F. Glover and D. Klingman, The generalized lattice-point problem, *Operations Res.*, 21 (1973) 141–155.

[8] F. Glover, D. Klingman and J. Stutz, The disjunctive facet problem: Formulation and solution techniques, *Operations Res.*, 22 (1974) 582–601.

[9] P.G. McKeown and D.S. Rubin, Neighboring vertices on transportation polytopes, to appear in *Naval Res. Logistics Quarterly*, 22 (1975) 365–374.

[10] D.S. Rubin, Vertex generation and cardinality constrained linear programs, *Operations Res.*, 23 (1975) 555–565.

[11] D.S. Rubin, Vertex generation and linear complementarity problems, Technical Report No. 74-2, Curriculum in Operations Research, University of North Carolina at Chapel Hill, December 1974.

[12] K. Tanahashi and D. Luenberger, Cardinality-constrained linear programming, Stanford University, 1971.

Annals of Discrete Mathematics 1 (1977) 467-477
© North-Holland Publishing Company

SENSITIVITY ANALYSIS IN INTEGER PROGRAMMING*

Jeremy F. SHAPIRO

Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

This paper uses an IP duality theory recently developed by the authors and others to derive sensitivity analysis tests for IP problems. Results are obtained for cost, right hand side and matrix coefficient variation.

1. Introduction

A major reason for the widespread use of LP models is the existence of simple procedures for performing sensitivity analyses. These procedures rely heavily on LP duality theory and the interpretation it provides of the simplex method. Recent research has provided a finitely convergent IP duality theory which can be used to derive similar procedures for IP sensitivity analyses (Bell and Shapiro [3]; see also Bell [1], Bell and Fisher [2], Fisher and Shapiro [6], Fisher, Northup and Shapiro [7], Shapiro [18]). The IP duality theory is a constructive method for generating a sequence of increasingly strong dual problems to a given IP problem terminating with a dual producing an optimal solution to the given IP problem. Preliminary computational experience with the IP dual methods has been promising and is reported in [7]. From a practical point of view, however, it may not be possible when trying to solve a given IP problem to pursue the constructive procedure as far as the IP dual problem which solves the given problem. The practical solution to this difficulty is to imbed the use of IP duality theory in a branch and bound approach (see [7]).

The IP problem we will study is

$$
\begin{aligned}
&v = \min cx \\
(1)\qquad &\text{s.t. } Ax + Is = b \\
&x_j = 0 \text{ or } 1, \quad s_i = 0, 1, 2, \dots, U_i,
\end{aligned}
$$

where A is an $m \times n$ integer matrix with coefficients a_{ij} and columns a_j, b is an $m \times 1$ integer vector with components b_i and c is a $1 \times n$ real vector with components c_j. For future reference, let $F = \{x^r, s^r\}_{r=1}^{R}$ denote the set of all feasible solutions to (1).

* Supported in part by the U.S. Army Research Office (Durham) under Contract No. DAHC04-73-C-0032.

We have chosen to add the slack variables explicitly to (1) because they behave in a somewhat unusual manner unlike the behavior of slack variables in LP. Suppose for the moment that we relax the integrality constraints in problem (1); that is, we allow $0 \le x_j \le 1$ and $0 \le s_i \le U_i$. Let u_i^* denote an optimal dual variable for the ith constraint in this LP, and let s_i^* denote an optimal value of the slack. By LP complementary slackness, we have $u_i^* < 0$ implies $s_i^* = 0$ and $u_i^* > 0$ implies $s_i^* = U_i$. In the LP relaxation of (1), it is possible that $0 < s_i^* < U_i$ only if $u_i^* = 0$. On the other hand, in IP we may have a non-zero price u_i^* and $0 < s_i^* < U_i$ because the discrete nature of the IP problem makes it impossible for scarce resources to be exactly consumed. Specific mathematical results about this phenomenon will be given in Section 2.

2. Review of IP duality theory

A dual problem to (1) is constructed by reformulating it as follows. Let G be any finite abelian group with the representation

$$G = Z_{q_1} \oplus Z_{q_2} \oplus \cdots \oplus Z_{q_r}$$

where the positive integers q_i satisfy $q_i \ge 2$, $q_i \mid q_{i+1}$, $i = 1, \ldots, r-1$, and Z_{q_i} is the cyclic group of order q_i. Let g denote the order of G; clearly $g = \prod_{i=1}^{r} q_i$ and we enumerate the elements as $\sigma_0, \sigma_1, \ldots, \sigma_{g-1}$ with $\sigma_0 = 0$. Let $\varepsilon_1, \ldots, \varepsilon_m$ be any elements of this group and for any m-vector f, define the element $\phi(f) = \sigma \in G$ by

$$\phi(f) = \sum_{i=1}^{m} f_i \varepsilon_i.$$

The mapping ϕ naturally partitions the space of integer m-vectors into g equivalence classes $S_0, S_1, \ldots, S_{g-1}$ where $f^1, f^2 \in S_K$ if and only if $\sigma_K = \phi(f^1) = \phi(f^2)$. The element σ_K of G is associated with the set S_K; that is, $\phi(f) = \sigma_K$ for all integer m-vectors $f \in S_K$.

It can easily be shown that (1) is equivalent to (has the same feasible region as)

(2a) $$v = \min cx,$$

(2b) $$\text{s.t. } Ax + Is = b,$$

(2c) $$\sum_{j=1}^{n} \alpha_j x_j + \sum_{i=1}^{m} \varepsilon_i s_i = \beta,$$

$$x_j = 0 \text{ or } 1,$$

(2d) $$s_i = 0, 1, 2, \ldots, U_i,$$

where $\alpha_j = \phi(a_j)$ and $\beta = \phi(b)$. The group equations (2c) are a system of r congruences and they can be viewed as an aggregation of the linear system $Ax + Is = b$. Hence the equivalence of (1) and (2). For future reference, let Y be the set of (x, s) solutions satisfying (2c) and (2d). Note that $F \subseteq Y$.

The IP dual problem induced by G is constructed by dualizing with respect to the constraints $Ax + Is = b$. Specifically, for each u define

$$(3) \qquad L(u) = ub + \min_{(x,s)\in Y} \{(c - uA)x - us\}.$$

The Langrangean minimization (3) can be performed in a matter of a few seconds or less for g up to 5000; see Glover [10], Gorry, Northup and Shapiro [11]. The ability to do this calculation quickly is essential to the efficacy of the IP dual methods. If g is larger than 5000, methods are available to try to circumvent the resulting numerical difficulty (Gorry, Shapiro and Wolsey [12]). However, there is no guarantee that these methods will work, and computational experience has shown that the best overall strategy is to combine these methods with branch and bound.

Sensitivity analysis on IP problem (1) depends to a large extent on sensitivity analysis with respect to the group G and the Langrangean L. Let

$$(4) \qquad \begin{aligned} g(\sigma; u) = \min \sum_{j=1}^{n} (c_j - ua_j)x_j + \sum_{i=1}^{m} -u_i s_i \\ \text{s.t.} \sum_{j=1}^{n} \alpha_j x_j + \sum_{i=1}^{m} \varepsilon_i s_i = \sigma, \\ x_j = 0 \text{ or } 1, \\ s_i = 0, 1, 2, \ldots, U_i. \end{aligned}$$

Then $L(u) = ub + g(\beta; u)$. Moreover, the algorithms in [10] and [11] can be used to compute $g(\sigma; u)$ for all $\sigma \in G$ without a significant increase in computation time.

It is well known and easily shown that the function L is concave, continuous and a lower bound on v. The IP dual problem is to find the greatest lower bound

$$(5) \qquad \begin{aligned} w = \max L(u) \\ \text{s.t. } u \in \mathbb{R}^m. \end{aligned}$$

If $w = +\infty$, then the IP problem (1) is infeasible.

The desired relation of the IP dual problem (5) to the primal IP problem (1) is summarized by the following:

Optimality Conditions: *The pair of solutions* $(x^*, s^*) \in Y$ *and* $u^* \in \mathbb{R}^m$ *is said to satisfy the optimality conditions if*

(i) $L(u^*) = u^*b + (c - u^*A)x^* - u^*s^*$,

(ii) $Ax^* + Is^* = b$.

It can easily be shown that a pair satisfying these conditions is optimal in the respective primal and dual problems. For a given IP dual problem, there is no guarantee that the optimality conditions can be established, but attention can be restricted to optimal dual solutions for which we try to find a complementary optimal primal solution. If the dual IP problem cannot be used to solve the primal problem, then $v > w$ and we say there is a *duality gap*; in this case, a stronger IP dual problem is constructed.

Specifically, solution of the IP problem (1) by dual methods is constructively achieved by generating a finite sequence of groups $\{G^k\}_{k=0}^{K}$, sets $\{Y^k\}_{k=0}^{K}$, and IP dual problems analogous to (5) with maximal objective function value w^k. The group $G^0 = Z_1$, $Y^0 = \{(x, s) \mid x_j = 0$ or 1, $s_i = 0, 1, 2, \ldots, U_i\}$ and the corresponding IP dual problem can be shown to be the linear programming relaxation of (1). The groups here have the property that G^k is a subgroup of G^{k+1}, implying directly that $Y^{k+1} \subset Y^k$ and therefore that $v \geq w^{k+1} \geq w^k$. Sometimes we will refer to G^{k+1} as a supergroup of G^k.

The critical step in this approach to solving the IP problem (1) is that if an optimal solution to the k^{th} dual does not yield an optimal integer solution, then we are able to construct the supergroup G^{k+1} so that $Y^{k+1} \subsetneq Y^k$. Moreover, the construction eliminates the infeasible IP solutions $(x, s) \in Y^k$ which are used in combination by the IP dual problem to produce a fractional solution to the optimality conditions. Since the set Y^0 is finite, the process must converge in a finite number of IP dual problem constructions to an IP dual problem yielding an optimal solution to (1) by the optimality conditions, or prove that (1) has no feasible solution. Details are given in [3].

The following theorem exposes how this IP duality theory extends the notion of complementary slackness to IP.

Theorem 1. *Suppose that $(x^*, s^*) \in Y$ and $u^* \in \mathbf{R}^m$ satisfy the optimality conditions. Then*

(i) *$u_i^* < 0$ and $s_i^* > 0$ implies $\varepsilon_i \neq 0$.*

(ii) *$u_i^* > 0$ and $s_i^* < U_i$ implies $\varepsilon_i \neq 0$.*

Proof. Suppose $u_i^* < 0$ and $s_i^* > 0$ but $\varepsilon_i = 0$. Recall that $(x^*, s^*) \in Y$ implies that $\sum_{j=1}^{n} \alpha_j x_j^* + \sum_{i=1}^{m} \varepsilon_i s_i^* = \beta$ and $L(u^*) = u^*b + (c - u^*A)x^* - u^*s^*$. Since $\varepsilon_i = 0$, we can reduce the value of s_i to 0 and still have a solution in Y. But this new solution to the Lagrangean has a cost of $L(u^*) + u_i^* s_i^* < L(u^*)$ contradicting the optimality of (x^*, s^*). The proof of case (ii) is similar. $\square$

The IP dual problem (5) is actually a large scale linear programming problem. Let $Y = \{x^t, s^t\}_{t=1}^{T}$ be an enumeration of Y. The LP formulation of (5) is

$$w = \max v$$

$$v \leq ub + (c - uA)x^t - us^t \qquad (6)$$

$$t = 1, \ldots, T$$

The linear programming dual to (6) is

$$w = \min \sum_{t=1}^{T} (cx^t) w_t$$

$$(7) \qquad \text{s.t.} \sum_{t=1}^{T} (Ax^t + Is^t) w_t = b,$$

$$\sum_{t=1}^{T} w_t = 1,$$

$$w_t \geq 0.$$

The number of rows T in (6), or columns in (7), is enormous. The solution methods given in Fisher and Shapiro [6] generate columns as needed by ascent algorithms for solving (6) and (7) as a primal dual pair. The columns are generated by solving the Lagrangean problem (5).

The formulation (7) of the IP dual has a convex analysis interpretation. Specifically, the feasible region in (7) corresponds to

$$\{(x, s) \mid Ax + Is = b, 0 \leq x \leq 1, 0 \leq s \leq U\} \cap [Y]$$

where the left hand set is the feasible region of the LP relaxation of the IP problem (1) and "[ ]" denotes convex hull. Thus, in effect, the dual approach approximates the convex hull of the set of feasible integer points by the intersection of the LP feasible region with the polyhedron $[Y]$. When the IP dual problem (5) solves the IP problem (1), then $[Y]$ has cut away enough of the LP feasible region to approximate the convex hull of feasible integer solutions in a neighborhood of an optimal IP solution.

3. Sensitivity analysis of cost coefficients

Sensitivity analysis of cost coefficients is easier than sensitivity analysis of right hand side coefficients because the set F of feasible solutions remains unchanged. As described in the previous section, suppose we have constructed an IP dual problem for which the optimality conditions are satisfied by some pair $(x^*, s^*) \in Y$ and u^*.

The first question we wish to answer is

> In what range of values can c_j vary without changing the value of the zero-one variable x_j in the optimal solution (x^*, s^*)?

We answer this question by studying the effect of changing c_j on the Lagrangean.

Theorem 2. *Let (x^*, s^*) and u^* denote optimal solutions to the primal and dual IP problems, respectively, satisfying the optimality conditions. Suppose the zero-one variable $x_j^* = 0$ and we consider varying its cost coefficient c_j to $c_j - \Delta c_j$. Then (x^*, s^*) remains optimal if*

(8) $\qquad \Delta c_i \geq \min\{0,\, g(\beta;u^*) - (c_i - u^*a_i) - g(\beta - a_i;u^*)\}$,

where $g(\cdot, u^*)$ *is defined in* (4).

Proof. Clearly, if $x_i^* = 0$ and (8) indicates that $\Delta c_i \geq 0$, or c_i increases, then x^* remains optimal with $x_i^* = 0$. Thus we need consider only the case when $g(\beta;u^*) - (c_i - u^*a_i) - g(\beta - a_i;u^*) < 0$. Let $g(\beta;u^* \mid x_i = k;\Delta c_i)$ denote the minimal cost in (4) if we are constrained to set $x_i = k$ when the change in the ith cost coefficient is Δc_i. If Δc_i satisfies (8), then

$$
\begin{aligned}
(9) \qquad g(\beta;u^* \mid x_i = 1;\Delta c_i) &= c_i + \Delta c_i - u^*a_i + g(\beta - a_i;u^* \mid x_i = 0;\Delta c_i)\\
&= c_i + \Delta c_i - u^*a_i + g(\beta - a_i;u^* \mid x_i = 0;0)\\
&\geq c_i + \Delta c_i - u^*a_i + g(\beta - a_i;u^*)\\
&\geq g(\beta;u^*)\\
&= g(\beta;u^* \mid x_i = 0;0)\\
&= g(\beta;u^* \mid x_i = 0;\Delta c_i),
\end{aligned}
$$

where the first equality follows from the definition of $g(\beta;u^* \mid x_i = 1;\Delta c_i)$, the second equality because the value of Δc_i is of no consequence if $x_i = 0$, the first inequality because $g(\beta - a_i;u^*)$ may or may not be achieved with $x_i = 0$, the second inequality by our assumption that (8) holds, the third equality because $g(\beta;u^*) = (c - u^*A)x^* + u^*s^*$ and $x_i^* = 0$, and the final equality by the same reasoning as the second equality. Thus, as long as Δc_i satisfies (8), it is less costly to set $x_i = 0$ rather than $x_i = 1$. ∎

On the other hand, marginal analysis when $x_i^* = 1$ is not as easy because the variable is used in achieving the minimal value $g(\beta;u^*)$. Clearly x^* remains optimal if c_i is decreased. As c_i is increased, x_i should eventually be set to zero unless it is uniquely required for feasibility.

Theorem 3. *Let* (x^*, s^*) *and* u^* *denote optimal solutions to the primal and dual IP problems, respectively, satisfying the optimality conditions. Suppose the zero-one variable* $x_i^* = 1$ *and we consider varying its cost coefficient* c_i *to* $c_i + \Delta c_i$. *Then* (x^*, s^*) *is not optimal in the Lagrangean if*

(10) $\qquad \Delta c_i > \min\{c_j - u^*a_j,\ j \in J(\alpha_i)\ \text{and}\ x_j^* = 0\} - (c_i - u^*a_i)$

where

$$J(\alpha_i) = \{j \mid \phi(a_j) = \phi(a_i) = \alpha_i\}$$

We assume there is at least one $x_j^* = 0$ *for* $j \in J(\alpha_i)$ *because otherwise the result is meaningless.*

Proof. Note that we can order the elements $j_1, j_2, \ldots, j_K$ in $J(\alpha_i)$ by increasing cost $c_j - u^* a_j$ with respect to u^* such that $x_j^* = 1$, $j = j_1, \ldots, j_k$, $x_j^* = 0$, $j = j_{k+1}, \ldots, j_K$. This is because all these variables x_j have the same effect in the constraints in (4). By assumption, there is an $x_l^* = 0$, and if $c_l - \Delta c_l - u^* a_l < c_j - u^* a_j$, then $x_l = 1$ will be preferred to $x_j = 1$ in (4). In this case, (x^*, s^*) is no longer optimal in the Lagrangean and the optimality conditions are destroyed. □

The inequality (10) can be a gross overstatement of when (x^*, s^*) ceases to be optimal in the Lagrangean. Systematic solution of $g(\beta; u^*)$ for increasing values of c_l is possible by the parametric methods we discuss next.

A more general question about cost coefficient variation in the IP problem (1) is the following

> How does the optimal solution change as the objective function c varies in the interval $[c^0, c^1]$?

Parametric IP analysis of this type has been studied by Nauss [15], but without the IP duality theory, and by the author in [21] in the context of multicriterion IP. We give some of the relevant results here. The work required to do parametric IP analysis is greater than the sensitivity analysis described above which is effectively marginal analysis.

For $\theta \in [0, 1]$ define the function

$$
\begin{aligned}
(11) \qquad v(\theta) = \min\,&((1-\theta)c^0 + \theta c^1)x, \\
&Ax + Is = b, \\
&x_j = 0 \text{ or } 1, \\
&s_i = 0, 1, 2, \ldots, U_i.
\end{aligned}
$$

It is easy to show that $v(\theta)$ is a piecewise linear concave function of θ. The IP dual objective function can be used to approximate $v(\theta)$ from below. Specifically, suppose (11) is solved by the IP dual at $\theta = 0$ and we consider increasing it. From (7), we have

$$
\begin{aligned}
&w(\theta) = \min \sum_{t=1}^{T} ((1-\theta)c^0 + \theta c^1)x^t w_t \\
(12) \qquad &\text{s.t.} \sum_{t=1}^{T} (Ax^t + Is^t) w_t = b \\
&\sum_{t=1}^{T} w_t = 1 \\
&w_t \geq 0
\end{aligned}
$$

where $w(\theta)$ is also a piecewise linear concave function of θ, and $w(0) = v(0)$ because we assume an IP dual has been constructed which solves the primal.

Without loss of generality, assume (x^1, s^1) is the optimal IP solution for $\theta = 0$. Then, $w_1 = 1$, $w_t = 0$, $t \geq 1$ is the optimal solution in the LP (12), and we can do parametric variation of $\theta \geq 0$ to find the maximal value, say θ^*, such that $v(\theta) = w(\theta)$ for all $\theta \in [0, \theta^*]$. The difficulty is that the number of columns in (12) is enormous. For this purpose, generalized linear programming can be used to generate columns as needed for the parametric analysis. Included is the possibility that when $w_1 = 1$, $w_t = 0$, $t \geq 1$, becomes non-optimal, another feasible IP solution will become the optimal solution in (12).

When a sufficiently large θ is reached such that $v(\theta) - w(\theta) > 0$, then we can use the iterative IP dual analysis described in section two to strengthen the dual and ultimately eliminate the gap. Numerical excesses may make this impractical. However, any IP dual can be used to reduce the work of branch and bound in the parametric analysis of the interval $[c^0, c^1]$ being studied. These IP duals give stronger lower bounds than the LP and related lower bounds used in [28].

4. Sensitivity analysis of right-hand side coefficients

This is a rich area of research which needs continuing investigation. Nevertheless, we can report on some results already obtained. Again we suppose that the IP duality theory has yielded an IP dual problem for which the optimality conditions hold for $(x^*, s^*) \in Y$, $u^* \in R^m$. As in LP, constraint i is not binding if $u_i^* = 0$. Specifically, it can easily be shown that x^* is optimal in IP(1) with the right hand side b_i equal to any of the numbers

$$\sum_{j=1}^{n} a_{ij}x_j^*, \quad \sum_{j=1}^{n} a_{ij}x_j^* + 1, \ldots, \sum_{j=1}^{n} a_{ij}x_j^* + U_i$$

for any row i with $u_i^* = 0$. The optimal value of the slack variable on such a row is $b_i - \sum_{j=1}^{n} a_{ij}x_j^*$.

To study further the effects of varying b, we define the perturbation function for b in a finite set B

$$(13) \qquad \begin{aligned} v(b) &= \min cx \\ \text{s.t. } Ax - Is &= b \\ x_j &= 0 \text{ or } 1 \\ s_i &= 0, 1, 2, \ldots, U_i. \end{aligned}$$

Attention is limited to a finite set rather than all integer vectors in R^m because a finite set is more likely to be the type of interest, and also because it avoids troublesome technical difficulties. For a finite set of integer right hand sides, universal upper bounds on the slacks can be found and used. The function $v(b)$ is poorly behaved, except for the property that $b^1 \geq b$ implies $v(b^1) \geq v(b)$, which makes it difficult to study. Note also that v is defined only on the integers and it is not differentiable, unlike perturbation functions in nonlinear programming.

For each right hand side $b \in B$, the finitely convergent duality theory given in [3] produces an IP dual problem which solves (13) by establishing the optimality conditions. These IP duals are related but specific results about their relationship are difficult to obtain. Instead, we consider the IP dual which solves (13) with the given right hand side b^0, and investigate its properties with respect to the other $b \in B$.

Let G denote the group used in the construction of the IP dual solving (13) with $b = b^0$. This group induces a family of g related dual problems defined on each of the g equivalence classes of right hand sides $S_0, S_1, \ldots, S_{g-1}$. For $b \in S_\sigma$ we have

(14)
$$w_\sigma(b) = \max L_\sigma(u) \quad \text{s.t. } u \in \mathbb{R}^m_+,$$

where

$$L_\sigma(u) = ub + g(\sigma; u).$$

By assumption

$$v(b^0) = cx^* = w_{\beta^0}(b^0) = L_{\beta^0}(u^*),$$

where $\beta^0 = \phi(b^0)$. The solution of problem (4) for all right hand sides σ gives us optimal solutions to a number of other IP problems (13). Let $(x(\sigma), s(\sigma))$ denote the optimal solution to (4) with right hand side σ when $u = u^*$. It is easy to show by direct appeal to the optimality conditions that $(x(\sigma), s(\sigma))$ is optimal in (13) with

$$b_i = \sum_{j=1}^{n} a_{ij}x_j(\sigma) + s_i(\sigma), \quad i = 1, \ldots, m.$$

Moreover, for i such that $u_i^* = 0$, $x(\sigma)$ and the corresponding slack values are optimal in (13) with

$$b_i = \sum_{j=1}^{n} a_{ij}x_j(\sigma), \ldots, \sum_{j=1}^{n} a_{ij}x_j(\sigma) + U_i.$$

Thus, we may immediately have optimal solutions to some of the IP problems (13) for $b \in B$. In addition, $x(\sigma)$ is a feasible but possibly non-optimal solution in (13) with $b \in B$ if $Ax(\sigma) \leq b$ and

$$b_i - \sum_{j=1}^{n} a_{ij}x_j(\sigma) \in \{0, 1, \ldots, U_i\}.$$

Note, however, that not all constraints of an IP problem can be allowed to vary parametrically. For example, a constraint of the form $x_{11} + x_{12} \leq 1$ indicating that project 1 can be started in period 1 or period 2, but not both, makes no sense when extended to the constraint $x_{11} + x_{12} \leq 2$. Some constraints of this type can be included in the Lagrangean calculation.

Marsten and Morin [14] have devised schemes for parametric analysis of the right hand side of IP problems. The duality results here can be integrated with their approach to provide tighter lower bounds for the branch and bound procedure. They consider b to be a real vector and observe jumps in the function $z(b)$. The selection of integer data for A and b in effect limits attention to the points where $z(b)$ might change value.

5. Sensitivity analysis of matrix coefficients

This analysis is similar to the cost coefficient analysis since the dual approach is to convert constraints to costs. The question we address is:

> In what range of values can a coefficient a_{il} vary without changing the value of the zero-one variables x_l in the optimal solution (x^*, s^*)?

As before, the answer to this question is easier if $x_l^* = 0$.

Theorem 4. *Let (x^*, s^*) and u^* denote optimal solutions to the primal and dual IP problems, respectively, satisfying the optimality conditions. Suppose the zero-one variable $x_l^* = 0$ and we consider varying the coefficient a_{il} to $a_{il} + \Delta a_{il}$ where Δa_{il} is integer. Then x^* remains optimal if Δa_{il} satisfies*

$$(15) \qquad -u_i^* \Delta a_{il} \geq \min\{0, g(\beta; u^*) - (c_l - u^* a_l) - g(\beta - \alpha_l - \Delta a_{il}\varepsilon_i; u^*)\}.$$

There is no restriction on Δa_{il} if $u_i^* = 0$.

Proof. The proof is identical to the proof of Theorem 2. The change Δa_{il} causes the change $-u_i^* \Delta a_{il}$ in the cost coefficient analogous to the change Δc_l in Theorem 2, and the group identity of a_l is changed to $\alpha_l + \Delta a_{il}\varepsilon_i$. □

A result similar to Theorem 3 for the case when $x_l^* = 1$ can be obtained. We omit further details. A more general type of IP matrix coefficient variation is the problem of IP column generation. Such a problem would arise, for example, if there were a subproblem to be solved whose solution provided a candidate column a with cost coefficient c to be added to IP(1). A construction to do this using the IP duality theory appears possible but will not be developed here.

6. Conclusions

We have presented some results for performing IP sensitivity analyses using IP duality theory. More research into these methods is needed, particularly a more extensive study of the family of IP dual problems which result as the right hand side in (1) is varied. Computational experience with these methods, in conjunction with branch and bound, is crucial and will suggest important areas of research. We mention that the work of Burdet and Johnson [5] appears to provide an analytic formalism for combining duality and branch and bound. Finally, the IP duality theory has been extended to mixed IP in [20] indicating that the results here can be readily extended to sensitivity analysis for mixed IP.

References

[1] D.E. Bell, Improved bounds for integer programs: A supergroup approach, IIASA RM-73-5, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1973, to appear in *SIAM J. Appl. Math.*

[2] D.E. Bell and M.L. Fisher, Improved integer programming bounds using intersections of corner polyhedra, *Math. Programming* 8 (1975) 345–368.

[3] D.E. Bell and J.F. Shapiro, A convergent duality theory for integer programming, RM 75-33, International Institute for Applied Systems Analysis, Schloss Laxenburg, Austria, 1975, to appear in *Operations Research*, 1977.

[4] C. Burdet and E.L. Johnson, A subadditive approach to the group problem of integer programming, *Math. Programming Study* 2 (1974) 51–71.

[5] C.A. Burdet and E.L. Johnson, A subadditive approach to solve linear integer programs, 1975.

[6] M.L. Fisher and J.F. Shapiro, Constructive duality in integer programming, *SIAM J. Appl. Math.* 27 (1974) 31–52.

[7] M.L. Fisher, W.D. Northup and J.F. Shapiro, Using duality to solve discrete optimization problems: Theory and computational experience, *Math. Programming Study* 3 (1975) 56–94.

[8] A.M. Geoffrion, Lagrangean relaxation for integer programming, *Mathematical Programming Study* 2 (1974) 82–114.

[9] A.M. Geoffrion and R.E. Marsten, Integer programming algorithms: A framework and state-of-the-art survey, *Management Science* 18 (1972) 465–491.

[10] F. Glover, Integer programming over a finite additive group, *SIAM J. Control* 7 (1969) 213–231.

[11] G.A. Gorry, W.D. Northup and J.F. Shapiro, Computational experience with a group theoretic integer programming algorithm, *Mathematical Programming* 4 (1973) 171–192.

[12] G.A. Gorry, J.F. Shapiro and L.A. Wolsey, Relaxation methods for pure and mixed integer programming problems, *Management Sci.* 18 (1972) 229–239.

[13] M. Held and R.M. Karp, The travelling-salesman problem and minimum spanning trees: Part II, *Math. Programming* 1 (1971) 6–25.

[14] R.E. Marsten and T.L. Morin, Parametric integer programming: The right-hand side case, 1975.

[15] Robert M. Nauss, Parametric integer programming, UCLA Working Paper No. 226, Western Management Science Institute, UCLA (1975).

[16] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).

[17] G.M. Roodman, Postoptimality analysis in zero-one programming by implicit enumeration, *Naval Research Logistic Quarterly* 19 (1972) 435–447.

[18] J.F. Shapiro, Generalized Lagrange multipliers in integer programming, *Operations Res.* 19 (1971) 68–76.

[19] J.F. Shapiro, A decomposition algorithm for integer programming problems with many columns, *Proc. 25th Annual ACM Conference*, Boston, August, 1972, pp. 528–533.

[20] J.F. Shapiro, A new duality theory for mixed integer programming, Operations Research Center Working Paper No. OR 033-74, Massachusetts Institute of Technology, 1974. To appear in *Proc. Congress on Math. Programming and its Applications*, Rome.

[21] J.F. Shapiro, Multiple criteria public investment decision making by mixed integer programming, pp. 170–182 in Multiple Criteria Decision Making, edited by H. Thiriez and S. Zionts, Springer-Verlag, 1976.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 479–494
© North-Holland Publishing Company

A LIFO IMPLICIT ENUMERATION SEARCH ALGORITHM FOR THE SYMMETRIC TRAVELING SALESMAN PROBLEM USING HELD AND KARP'S 1-TREE RELAXATION*

T.H.C. SMITH

Department of Statistics, Rand Afrikaans University, Johannesburg, R.S.A.

G.L. THOMPSON

Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

We propose here a LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem which uses the 1-tree relaxation of Held and Karp. The proposed algorithm has significantly smaller memory requirements than Held and Karp's branch and bound algorithm. Computational experience with this algorithm and an improved version of Held and Karp's algorithm is reported and on the basis of the sample it can be stated that the proposed algorithm is faster and generates many fewer subproblems than Held and Karp's algorithm.

1. Introduction

In two excellent papers, [10] and [11], Held and Karp investigated the relationship between the symmetric traveling salesman problem and the minimum spanning tree problem. They used this relationship to determine a lower bound on the length of a minimal tour and in [11] developed an efficient ascent method for improving this lower bound. They incorporated this method in a branch-and-bound algorithm for the solution of the symmetric traveling salesman problem and reported exceptionally good computational experience with this algorithm. In a subsequent paper [12] Held, Wolfe and Crowder reported additional computational experience with a refinement of the above ascent method used in Held and Karp's branch-and-bound algorithm, verifying the effectiveness of the method in obtaining a near-maximal lower bound (of this type) on the minimal tour length. It is worth noting that Cristofides [2] independently considered an ascent method for the problem of finding a shortest Hamiltonian chain.

A major disadvantage of Held and Karp's (breadth-first) branch-and-bound algorithm (and of other branch-and-bound algorithms) is the creation of a list (of unpredictable length) of subproblems for each of which certain information must be

* This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract N00014-75-C-0621 NR 047-048 with the U.S. Office of Naval Research.

kept. The memory requirements of such large lists severely limit the sizes of problems that can be solved using only the high speed memory of a computer.

We propose here a LIFO (depth first) implicit enumeration algorithm [1, 5, 17, 23] for the solution of the symmetric traveling salesman problem which does not suffer from this memory disadvantage and which, on the basis of some limited computational experience, performs better than an improved version of Held and Karp's branch-and-bound algorithm.

2. Terminology and review

Let G be a complete undirected graph with node set $N = \{1, 2, \ldots, n\}$. A *cycle* C in G is a connected subgraph of G in which each node is met by exactly two edges. If (N_1, N_2) is a nontrivial partition of N, then the nonempty set of edges (i, j), $i \in N_1$, $j \in N_2$, of G is called a *cutset* in G. A *spanning tree* T in G is a connected subgraph of G with node set N which contains no cycles. The edges of G in T are called *branches* of T while all other edges of G are called *chords* of T. The *fundamental cycle* of a chord c is the set of edges in the unique cycle in G formed by c and a subset of the branches. The *fundamental cutset* of a branch b is the set of edges in the cutset on the partition defined by the two connected subgraphs of G which are formed when b is removed from T.

Suppose T_1 and T_2 are two spanning trees in G such that exactly one branch b_0 of T_1 is a chord of T_2 (and exactly one chord e_0 of T_1 is a branch of T_2). For any branch b of T_i let $D_i(b)$ be its fundamental cutset and for any chord c of T_i let $C_i(c)$ be its fundamental cycle, $i = 1$ or 2. The following theorem, which is also a consequence of Proposition 2 in [24], relates the fundamental cycles and cutsets of T_1 and T_2.

Theorem 1. *Let Δ denote the symmetric difference of two sets. Then we have:*

(i) $C_2(b_0) = C_1(e_0)$ *and* $D_2(e_0) = D_1(b_0)$;

(ii) *if $c \neq e_0$ is a chord of T_1, it is also a chord of T_2 and if $b_0 \notin C_1(c)$, then $C_2(c) = C_1(c)$, else $C_2(c) = C_1(c) \Delta C_1(e_0)$;*

(iii) *if $b \neq b_0$ is a branch of T_1, it is also a branch of T_2 and if $e_0 \notin D_1(b)$, then $D_2(b) = D_1(b)$, else $D_2(b) = D_1(b) \Delta D_1(b_0)$.*

A proof of this theorem can easily be constructed by drawing two trees satisfying the hypothesis of the theorem.

Assume each edge (i, j), $i \in N$, $j \in N$, of G has an associated length c_{ij}. For any subset S of edges of G, the total length equals $\sum_{(i,j) \in S} c_{ij}$. The *minimal spanning tree problem* is that of finding a spanning tree T of G with minimum total length of the set of edges in T. Several methods for solving this problem have been proposed (see [3, 16, 19, 21, 24]). According to the computational experience reported in [15], the most efficient of these in the case of a complete graph is the algorithm of Prim and Dijkstra.

The following are well-known necessary and sufficient conditions for a minimal spanning tree ([3a, p. 175] and [24]).

NS1. A spanning tree T is minimal if and only if every branch of T is at least as short as any chord in its fundamental cutset.

NS2. A spanning tree T is minimal if and only if every chord of T is at least as long as any branch in its fundamental cycle.

Part (ii) of Theorem 2 also appears in [24].

Theorem 2. *Suppose T_1 is a minimal spanning tree.*

(i) *If the length of chord c_s of T_1 is made arbitrarily small in order to force c_s into the minimal spanning tree, a new minimal spanning tree T_2 can be obtained from T_1 by exchanging c_s and a longest branch b_t in its fundamental cycle.*

(ii) *If the length of a branch b_t of T_1 is made arbitrarily large in order to force b_t out of the minimal spanning tree, a new minimal spanning tree T_2 can be obtained from T_1 by exchanging b_t and a shortest chord c_s in its fundamental cutset. In both cases the increase in the length of the minimal spanning tree equals the length of c_s minus the length of b_t.*

The proof is easy and is omitted.

Let G' be the complete subgraph of G with node set $N' = N - \{1\}$. A *1-tree* T in G is a spanning subgraph of G containing two edges incident to node 1 as well as the edges of a spanning tree T' in G'. The edges of T will also be referred to as *branches* and the edges of G not in T as *chords*. When we refer to the fundamental cutset/cycle of a branch/chord, we implicitly assume that it is an edge of G'. The *minimal 1-tree problem* is then the problem of finding the shortest two edges incident to node 1 as well as a minimal spanning tree in G'.

The *traveling salesman problem* is that of finding a minimal tour (i.e., a 1-tree with exactly two branches meeting each node in N). As noted by Held and Karp in [10] and Christofides [2], if, for any set of node weights $\{\pi_i, i \in N\}$, we transform the edge lengths using the transformation $c'_{ij} = c_{ij} + \pi_i + \pi_j$, $i \in N$, $j \in N$, the set of minimal tours stays the same while the set of minimal 1-trees may change.

As indicated in these references, the lengths of these minimal 1-trees can be used to construct lower bounds for tour lengths, which are usefull in the branch and bound search.

3. Ascent methods

In [10] Held and Karp gave, among others, an ascent method which iteratively increases the lower bound L by changing a single node weight at each iteration. In a second paper [11] Held and Karp proposed a more efficient method for finding a set of node weights which yield a good lower bound. They implemented this method (in a rather crude way) in another branch-and-bound algorithm (which we will henceforth call the HK-algorithm) for the solution of the symmetric traveling salesman problem and obtained excellent computational results.

In a subsequent paper [12] Held, Wolfe and Crowder reported additional computational experience with a refined implementation of Held and Karp's ascent method, verifying the effectiveness of the method in obtaining a near-maximal lower bound (of the type considered) on the minimal tourlength. A single iteration of this ascent method can be described as follows:

Given a set of node weights $\{\pi_i, i \in N\}$ and an upper bound U on the minimal tourlength, find a minimal 1-tree T with respect to the transformed edge lengths and let L be the lower bound computed from T. If T is a tour the ascent is terminated since L is the optimal lower bound. Otherwise let d_i be the number of branches meeting node i and λ be a given positive scalar smaller than or equal to 2. Compute the scalar quantity $t = \lambda(U - L)/\sum_{i \in N}(d_i - 2)^2$ and replace the old set of node weights with the new set of node weights $\{\pi'_i, i \in N\}$ computed from the following formulas:

$$\pi'_i = \pi_i + t(d_i - 2), \quad i \in N. \tag{1}$$

Our implementation of the ascent method is based on the strategies used in [11] and [12]. It requires input parameters K, z, α, β, τ and λ, where K is the initial number and z the minimum number of ascent iterations, and α, β, τ and λ are tolerances. Given a set of node weights and a upper bound U on the minimal tourlength, we initially do K ascent iterations of the type indicated above with the given tolerance value of λ used in (1). Thereafter we successively halve λ, put K = maximum $(K/2, z)$ and do another K ascent iterations until the first iteration at which at least one of the following statements is true (at which point the ascent is terminated):

(i) the computed t value is less than the tolerance α,

(ii) the minimal 1-tree is a tour,

(iii) K has the value z and no improvement in the (maximum) lower bound of at least β occurred in a block of $4z$ ascent iterations,

(iv) $U - L \leq \tau$.

At termination of an ascent we restore the set of node weights which yielded the current lower bound and compute a minimal 1-tree with respect to the transformed edge lengths. The particular values of the tolerances α and β (see (i) and (iii) above) that we used in our computational work, are given in the section on computational results. The tolerance τ used in (iv) should be zero in general but under the assumption that the original edge lengths are integers, τ can be taken as a real number smaller than unity. In our code for the improved version of the HK-algorithm (which we henceforth call the IHK-algorithm) we took $\tau = 0.999$. Furthermore we took the quantity z (which Held, Wolfe and Crowder call a "threshold value") equal to the integer part of $n/8$. The initial value of λ in an ascent was taken equal to 2, except where noted otherwise.

In the HK-algorithm one can distinguish between the use of the ascent method on the original problem (called the initial ascent) and its use on subproblems generated subsequently in the branching process (called general ascents). In the HK-algorithm the initial and general ascents are done in exactly the same way. In the HKJ-algorithm we implemented the initial and general ascents slightly differently, starting the initial ascent with $K = n$ but any general ascent with $K = z$. This had the effect that a general ascent generally required fewer ascent iterations than the initial ascent. Intuitively this is correct if one reasons that if the initial ascent finds a good set of node weights, a general ascent should require fewer ascent iterations than the initial ascent to find a good set of node weights for the subproblem under consideration.

In the HKJ-algorithm we used the same branching strategy as used by Held and Karp in their HK-algorithm. We noted that a last-created subproblem in a branching was often a subproblem with least lower bound among the subproblems currently in the list and hence could automatically be selected as the next subproblem to be subjected to the general ascent and subsequent branching. Our computational experience showed that the ascent method almost never produced an increase in the lower bound for a subproblem of this kind. We eliminated the ascent for such a subproblem in our code for HKJ and in the three problems we used to test for an improvement, we found that the size of the search tree did not increase significantly but that the total number of ascent iterations (and hence total run time) dropped considerably. For instance in KT57, the 57-node problem of Karg and Thompson [14], the number of nodes in the search tree increased from 378 to 409 while the number of ascent-iterations dropped from 8744 to 4407, cutting total run time from 8.25 minutes to 4.50 minutes.

In any branch-and-bound or implicit enumeration algorithm for the traveling salesman problem it is important to have a good upper bound U on the minimal tourlength. We used the first phase of the heuristic algorithm of Karg and Thompson [14], incorporating most of the improvements given by Raymond [20], to find a reasonable value for U. This algorithm starts out with a subtour through a given pair of nodes. We took U as the minimum tourlength among the $(K+1)$ tours generated by successively starting out with a subtour through the node pairs $(1, 2)$, $(1, 7)$, $\ldots$, $(1, 5K+2)$ where K is the largest integer smaller than $(n-1)/5$.

4. A LIFO implicit enumeration algorithm

A major disadvantage of a breadth first branch-and-bound algorithm such as the HK-algorithm, is the creation of a list (of unpredictable length) of subproblems for each of which certain information must be kept in memory. We propose here a LIFO implicit enumeration search algorithm, which we henceforth call the IE-algorithm, for the solution of the symmetric traveling salesman problem which does not suffer from this disadvantage, using the ideas in [1, 5, 6, 17, 23]. A stepwise description of this algorithm follows:

Step 0 (Initialization). Let the current subproblem be the original problem. Compute an upper bound U on the minimal tourlength and go to step 1.

Step 1 (Calculation of a lower bound for the current subproblem). Apply the ascent to the current subproblem to obtain a lower bound L on the minimal tourlength. If the ascent terminates because the minimal 1-tree is a tour or because $U - L < \tau$, go to step 3. Otherwise go to step 2.

Step 2 (Partitioning of the current subproblem). Select a node in N' which is met by more than two branches of the current minimal 1-tree. Let S be the set of all branches incident to this node which are not fixed in while F is the set of all branches incident to this node which are fixed in. Go to (a).

(a) If $|S \cup F| \leq 2$, go to step 1. Otherwise remove the branch e with the longest transformed length from the set S and determine the increase ε in the lower bound if e would be fixed out as well as the chord c which should be exchanged with e to obtain a minimal 1-tree for the resulting subproblem (if e is not incident to node 1 use Theorem 2(ii), otherwise c is the shortest chord incident to node 1 and ε is the nonnegative difference in transformed lengths between e and c). If $U - L - \varepsilon \geq \tau$, go to (b). Otherwise go to (c) since fixing e out would cause the lower bound to exceed the upper bound for the resulting subproblem.

(b) Fix e out of the minimal 1-tree (by changing its length temporarily to a large number) and find the new minimal 1-tree by exchanging e and c. If the resulting 1-tree is a tour, go to step 3. Otherwise go to (a).

(c) Fix e in the minimal 1-tree (by changing its length temporarily to a small number). If either of the end nodes of e is now met by two fixed branches, go to step 4. Otherwise set $F = F \cup \{e\}$ and go to (a).

Step 3 (Backtrack and create new current subproblem).

(a) If there are no fixed edges, go to step 5. Otherwise free the last fixed edge e by restoring its length to its original value. If e is a branch, go to (b). Otherwise go to (c).

(b) If e is incident to node 1 and longer than the shortest chord c incident to node 1, exchange e and c to get a minimal 1-tree. Otherwise, if e is longer than the shortest chord c in its fundamental cutset, exchange e and c to get a minimal 1-tree (see Theorem 2(ii)). Go to (a).

(c) Determine the increase ε in the lower bound if e would be fixed into the minimal 1-tree as well as the branch b which should be exchanged with e to obtain a minimal 1-tree (if e is not incident to node 1, use Theorem 2(i), otherwise ε equals the difference in transformed lengths between e and the longest branch b incident to node 1). If $\varepsilon < 0$, exchange e and branch b to get the new minimal 1-tree. If $U - L \geq \tau$, go to (d). Otherwise go to (a) since fixing e in would cause the lower bound to exceed the upper bound for the resulting subproblem.

(d) If either of the endnodes of e is met by two fixed branches, go to (a) since e cannot also be fixed in the minimal 1-tree. Otherwise fix e in the minimal 1-tree and if e is still a chord, exchange e and the branch b to get the new minimal 1-tree. If either of the endnodes of e is now met by two fixed branches, go to step 4. Otherwise go to step 1.

Step 4 (Create new current subproblem by skipping).

(a) For each endnode of e met by two fixed branches, consider successively all nonfixed edges incident to this node. If the edge e' currently under consideration is a chord, fix it out. Otherwise determine, in the same way as in step 2(a), the increase ε in the lower bound if e' would be fixed out of the minimal 1-tree. If $U - L - \varepsilon \leq \tau$, go to step 3. Otherwise fix e' out of the minimal 1-tree and find the new minimal 1-tree by exchanging e' and the appropriate chord.

(b) Go to step 2.

Step 5 (Termination). The tour which yielded the current upper bound U solves the original traveling salesman problem.

We represented the 1-tree T in the FORTRAN V implementation of the IE-algorithm as the two nodes in N' connected to node 1 together with the underlying spanning of tree T' in G' which we represented as an arborescence, using the three index scheme of Johnson [13], augmented by the distance index of Srinivasan and Thompson [22]. Fundamental cutsets and cycles were found utilizing the ideas in [13] and [22]. The updating of the four-index representation after a branch-chord exchange (pivot) was handled by the method given in [7]. For a typical 60-node problem the mean times on the UNIVAC 1108 for:

(i) finding the shortest chord in a fundamental cutset was 15.9 milliseconds,

(ii) finding the longest branch in a fundamental cycle was 0.4 milliseconds,

(iii) updating the 1-tree representation after a branch-chord exchange was 0.8 milliseconds,

(iv) finding a minimal 1-tree using the Prim-Dijkstra algorithm was 61.5 milliseconds.

The ascent method used in the IE-algorithm was exactly the same as that used for the HK1-algorithm, as described in the previous section. The parameter τ used in the description of the IE-algorithm is the same as in the ascent method. We again assumed integer data and took $\tau = 0.9$ on all test problems except T46, for which we took $\tau = 0.999$.

5. Computational results

The computational comparison of the HK1- and IE-algorithms is based on a sample consisting of nineteen problems. Problems DF42 and KT57 are respectively 42-node and 57-node problems that appear in [14] while HK48 is the 48-node problem of [9]. Problem T46 is the 46-node Tutte problem given in [11] (we associated a length of zero with each edge of the graph on page 23 of [11] and a length of 1 with every edge of T_{46} which does not appear in the graph). The other fifteen problems were randomly generated as described below.

The input to the random problem generator consists of five parameters, I1 to I5. A rectangle is partitioned vertically into I1 blocks of height I4 and each of these blocks is partitioned horizontally into I2 blocks of breadth I4 with the result that the original rectangle with dimensions $I1 \times I4$ by $I2 \times I4$ is partitioned into $I1 \times I2$ square blocks with side length I4. Using a random number generator, I3 nodes are chosen randomly in each block. The output of the problem generator is the set of coordinates for the resulting $n = I1 \times I2 \times I3$ nodes generated. The distance matrices for these random problems were calculated using the Euclidean distance measure, rounded down to the next integer. The parameter values used for the different problems are given in Table 1. The actual sets of coordinates for each of these problems are available on request from the authors.

Table 1

Problems	I1	I2	I3	I4
R48–R485	3	[illegible]	4	500
R600	3	4	5	500
R601–R605	3	5	4	300
R606–R608	1	1	60	1500

The computational results of applying the HKI- and IE-algorithms to the above-mentioned nineteen problems are given in Table 2. The identification of the columns in Table 2 is as follows:

(1) Mean time in milliseconds to compute *one* near-optimal tour using the Karg–Thompson–Raymond algorithm.

(2) Mean time in milliseconds for one ascent iteration (see section on ascent methods).

(3) Upper bound U on the minimal tourlength found using the Karg–Thompson–Raymond algorithm.

(4) Lower bound L on the minimal tourlength after the initial ascent (the same for both algorithms).

(5) Minimal tourlength L^*.

(6) Number of subproblems generated by the HKI-algorithm which were never chosen as a subproblem of least lower bound.

(7) Number of subproblems chosen as a subproblem of least lower bound by the HKI-algorithm which did not lead to branching because of a lower bound exceeding the current upper bound U.

(8) Number of subproblems which lead to branching in the HKI-algorithm.

(9) Total number of ascent iterations required by the HKI-algorithm.

(10) Maximum number of subproblems on the storage list during computation (for the HKI-algorithm).

(11) Total number of subproblems generated by steps 2(b), 2(c) and 3(d) of the IE-algorithm.

(12) Total number of skipping steps (step 4) for the IE algorithm.

Table 2[b]

Problem	(1)	(2)	(3)	(4)	(5)	(6)	HKI (7)	(8)	(9)	(10)	IE (11)	(12)	(13)	(14)
DF43	308	31	699	696.0	699	0	12	45	401	13	6	0	192	5.8
T46	—	54	1	0.0	1	0	292	748	2916	102	148	28	2664	96.7
HK48	456	39	11511	11443.9	11461	21	11	35	571	31	6	0	234	9.4
K157	624	56	13012	12902.5	12955	58	100	251	4407	103	38	2	1409	61.6
R481[a]	454	40	9783	9547.0	9729	391	186	767	12773	391	346	37	9385	391.6
R482	411	40	10680	10661.4	10691	0	21	49	466	15	6	0	304	12.1
R483	421	39	10180	10174.5	10191	0	5	30	261	6	4	0	197	7.6
R484	437	39	9484	9417.4	9484	0	80	221	2039	49	12	0	539	21.2
R485	471	39	9844	9827.3	9844	0	10	42	298	10	6	0	265	10.5
R600	748	65	10475	10359.[illegible]	10374	33	9	34	399	43	37	0	1296	85.3
R601[a]	731	60	11757	11588.0	11816	254	155	545	12053	254	121	21	4061	246.5
R602	727	60	12011	11777.0	11777	0	0	0	128	0	0	0	173	7.7
R603[a]	737	57	12669	12578.6	12699	254	86	423	10079	254	222	24	7106	414.6
R604	725	61	12551	12482.4	12497	46	24	97	1514	56	16	0	675	41.8
R605[a]	725	60	12278	12161.8	12262	254	45	379	8131	254	343	28	10570	646.7
R606	701	57	8159	8070.9	8073	56	0	41	361	56	4	0	312	18.7
R607	704	59	8557	8514.4	8533	235	190	594	12849	191	125	17	3715	234.1
R608[a]	710	57	8795	8805.4	8803	254	44	295	8734	254	59	4	2319	139.2
R609	699	57	9590	9184.4	9156	29	245	587	6845	105	4	0	314	18.9

[a] HKI not completed because of insufficient storage. Lower bound for least lower bound subproblem on list at termination was 9678.2, 11653.1, 12621.1, 12198.2 and 8845.4 for R481, R601, R603, R605 and R608 respectively.

[b] In T46 we took $\alpha = \beta = 0.001$. In all other problems we took $\alpha = 0.01$ and $\beta = 0.2$

(13) Total number of ascent iterations required by the IE-algorithm.

(14) Total runtime in seconds for the IE-algorithm (exclusive of the time to compute an initial upper bound U).

All times reported were obtained on a Univac 1108.

When comparing the performance of the two algorithms, it is natural to compare their respective runtimes. However in both algorithms the major part of runtime is spent performing ascent iterations (in the case of the IE-algorithm more than 95% of the total runtime). Since the total number of ascent iterations does not depend on actual coding or on the particular computer used (as does total runtime), we consider this statistic a better measure of comparison than total runtime. As can be seen from the entries in columns (9) and (13) of Table II, the IE-algorithm required fewer ascent iterations than the HKI-algorithm for all problems solved by both algorithms except R600. Excluding the problems not solved by the HKI-algorithm (because of insufficient storage for all the subproblems generated) and problem R602 for which a tour was found in the initial ascent, the IE-algorithm required on the average seven ascent iterations for every ten ascent iterations required by the HKI-algorithm. We do report the total runtime for the IE-algorithm in column (14) of Table 2. A lower bound on the total runtime for the HKI-algorithm can be obtained by multiplying the number of ascent iterations with the mean time for an ascent iteration.

A second important statistic which does not depend on the actual coding or the particular computer used, is the total number of subproblems generated during computation. In the case of the HKI-algorithm this number is given by the sum of the entries in columns (6), (7) and (8) of Table 2 while for the IE-algorithm it is given by the entry in column (11) of Table 2. As can be seen from Table 2, IE generated fewer subproblems than HKI on all problems except R602 including the problems that could not be solved by HKI. On the average HKI generated more than eight times as many subproblems than IE, excluding the problems not solved by HKI and problem R602.

A third basis of comparison between the two algorithms is the total memory requirements. For a 60-node problem the total memory requirements for the IE-algorithm was $10K$ (where $K = 1024$) memory locations while the HKI-algorithm required $7K$ memory locations for everything except the list of subproblems. An additional $34K$ main storage locations and $128K$ external storage locations (on a drum) were reserved for this list. This memory allocation for the subproblem list may seem excessive but in fact five of the nineteen problems in the sample required more list storage than this.

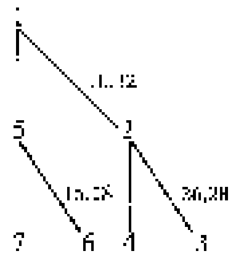
For every subproblem generated by HKI, the following information must be kept: (i) a set of node weights, (ii) the set of edges fixed in the minimal 1-tree, (iii) the set of edges fixed out of the minimal 1-tree and (iv) the cardinality of the sets in (ii) and (iii). In our implementation of the HKI-algorithm we packed the set of fixed edges so that a single memory location could contain information about three fixed edges. Therefore the total memory requirements for the information about a subproblem came to $n + n(n-1)/6 + 2$ memory locations for an n-node problem. For $n = 60$ this number equals 652 so that the $162K$ memory locations reserved for the list could accommodate 254 subproblems.

For five of the nineteen problems in our sample the HKI-algorithm generated more subproblems than could be accommodated in the $162K$ reserved memory locations. Since for a given problem, there is no reasonable upper bound on the number of subproblems to be generated by the HKI-algorithm (or the HK-algorithm), these unpredictable memory requirements are a serious disadvantage of both the HKI- and HK-algorithms.

We may also note the reason for the large number of subproblems being generated by both algorithms for problems R481, R601, R603, R605 and R608. On the basis of Held, Wolfe and Crowder's results [12] we are fairly confident that the lower bound L generated in the initial ascent was close to its optimal value. But in each of these problems the difference $L^* - L$ between the minimal tourlength and the lower bound L at the end of the initial ascent was much larger than the corresponding difference for the fourteen problems which generated many fewer subproblems. We suggest that this difference may therefore be a useful measure of problem difficulty.

An explanation for the fact that the IE-algorithm generates many fewer subproblems than the HKI-algorithm lies in the particular way subproblems are generated in step 2 of the IE-algorithm. The latter method of subproblem generation is much more oriented towards the goal of finding a minimal 1-tree that is a tour than is the partitioning method used in the HKI- and HK-algorithms. Our partitioning of a subproblem in step 2 of IE forces a minimal 1-tree towards a tour by fixing out "excess" branches of the minimal 1-tree. This involves the same idea as is present in the ascent method which can be viewed as a penalty method (see [2]) which forces the minimal 1-tree towards a tour by "penalizing" a node met by more than two branches (by increasing its node weight) and by "rewarding" a node met by only one branch (by decreasing its node weight).

In [11] Held and Karp presented the search trees for the problems for which they reported computational experience. It is interesting to compare their search trees for the problems DF42, HK48 and KT57 with the search trees generated by the IE algorithm for the same problems. These are represented respectively in Figs. 1, 2, and 3. The search trees for DF42 and HK48 correspond to the runs reported in Table 2 while the search tree for KT57 presented in Fig. 3 was obtained by starting each general ascent with the parameter λ set to 1 instead of 2. Note that, unlike Held and Karp's search trees which have some or all of the terminal nodes omitted, we show the *complete* search trees. The node numbers (underlined) in the search trees in Figs. 1, 2 and 3 correspond to the order in which the subproblems represented by the nodes were generated. If one views the search tree as a downward-directed arborescence with root node 1, the branch leaving a node vertically/obliquely represents an edge of G being fixed in/out with the end-nodes of the edge being fixed given next to the oblique branch.



Optimum tour found by the heuristic program

Fig. 1. DE42

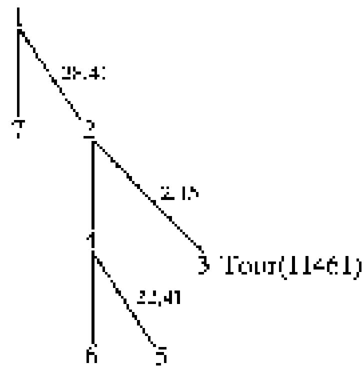


Fig. 2. HK48

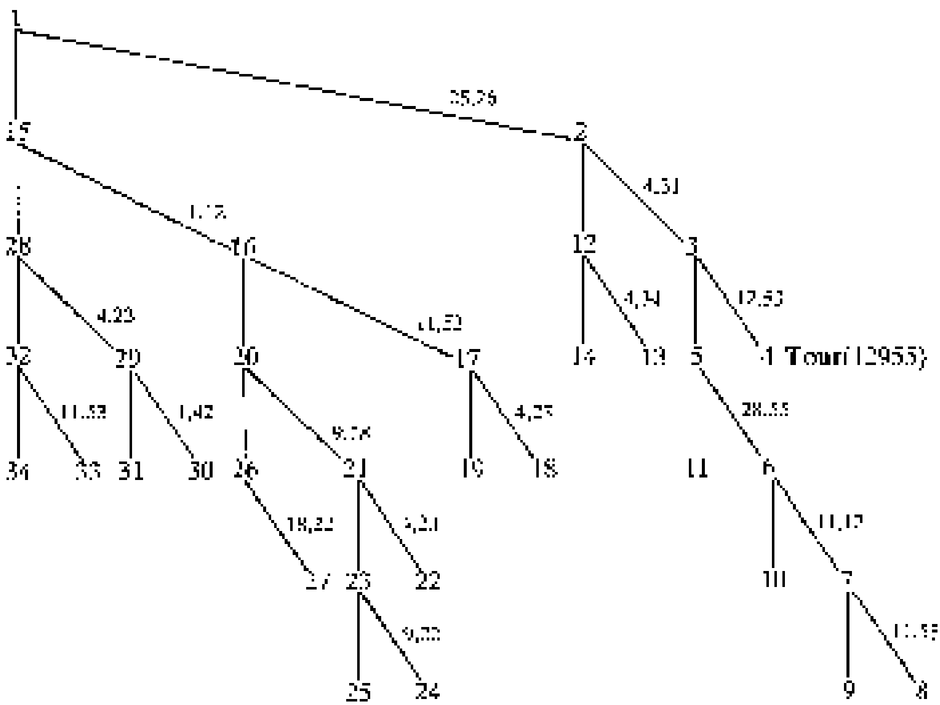


Fig. 3. KT57

After completing the experiments described above, we obtained the recent computational results of Hansen and Krarup [8]. They present an improved version of the HK-algorithm and report computational experience on an IBM 360/75 computer.

We generated three 15-problem samples of 50, 60 and 70 node problems each as well as five 80 node problems in the same manner as Hansen and Krarup and solved them with the IE-algorithm. Since the Karg–Thompson–Raymond heuristic cannot

Table 3*

$n = 50$				$n = 60$				$n = 70$				$n = 80$			
Gap	Iterations	Nodes	Time	Gap	Iterations	Nodes	Time	Gap	Iterations	Nodes	Time	Gap	Iterations	Nodes	Time
0.00	67	1	2.6	0.00	418	15	24.9	0.23	550	17	45.8	0.00	109	1	11.8
0.00	56	1	2.4	0.52	858	23	51.0	0.63	1417	39	115.7	0.31	695	11	75.4
0.00	104	2	4.3	0.00	65	1	3.8	0.00	82	1	6.6	0.17	2530	59	275.4
0.00	74	1	3.1	0.00	76	1	4.5	0.00	109		8.7	0.00	125	1	13.6
1.17	1863	70	79.5	0.26	1165	35	70.3	0.05	219	3	17.6	0.12	361	5	38.9
0.00	84	1	3.5	0.00	80	1	4.7	0.37	900	22	64.5				
0.00	60	1	2.5	0.24	341	9	20.2	0.10	191	3	15.4				
0.07	307	7	9.6	0.51	3211	65	190.9	0.00	66	1	5.3				
0.30	1114	39	46.7	0.10	977	35	58.0	0.18	2299	67	185.7				
0.41	651	21	27.7	0.00	80	1	4.7	0.39	2687	63	217.6				
0.08	894	27	37.3	0.09	873	33	48.9	0.43	1129	29	91.2				
0.02	936	27	38.9	0.10	195	5	11.5	0.00	80	1	6.4				
0.16	731	21	30.4	0.00	73	1	4.3	0.72	1104	25	89.7				
0.20	604	19	25.2	0.00	61	1	3.6	0.16	555	15	44.9				
0.10	452	15	18.8	0.06	169	3	10.0	0.00	117	1	9.4				
Average Runtime = 22.1				Average Runtime = 34.1				Average Runtime = 61.6				Average Runtime = 83.0			

* $\alpha = 0.01$, $\beta = 0.1$, $\gamma = 0.9$ in all problems.

be expected to provide a good upper bound U, for the type of problem under consideration, we took as upper bound 1.01 times the value of the lower bound at the end of the initial ascent (if no tour is found, the algorithm has to be run again with a higher upper bound). In the initial ascent we computed the stepsize as $t = \lambda(0.5M)/\sum_{i\in N}(d_i - 2)^2$ (where M is the maximum lower bound obtained in the current ascent) while in a general ascent we took $t = \lambda(0.005\,M)/\sum_{i\in N}(d_i - 2)^2$. All ascents were started with the parameter K set equal to z, the threshold value.

Our computational experience with the above fifty problems are reported in Table 3 where the column headings have the following interpretations:

Gap: The difference between the optimal tour length and the lower bound at the end of the initial ascent as a percentage of the optimal tourlength.

Iterations: The total number of ascent iterations.

Nodes: The total number of subproblems generated in steps 2(b), 2(c) and 3(d) of IE.

Time: The total runtime in seconds on the UNIVAC 1108.

A 100 node problem was also solved and took 13.6 minutes on the UNIVAC 1108, generating 95 nodes and requiring 5014 ascent iterations.

For the reasons stated above it is extremely difficult to compare the IE-algorithm with that of Hansen and Krarup. However, there does exist the possibility of improving the IE-algorithm further by making use of efficient sorting techniques and Kruskal's algorithm for finding a minimal spanning tree (see [16]) as done by Hansen and Krarup.

6. Conclusions

Our computational results indicate that the IE-algorithm is considerably faster than the HKI-algorithm. Since the major computational effort in the IE-algorithm is spent on Step 1 (the ascent method) in order to find good lower bounds on subproblems, an increase in the efficiency of the algorithm can be obtained by speeding up the ascent method. We are currently considering techniques for doing the latter.

References

[1] E. Balas, An additive algorithm for solving linear programs with zero-one variables, *Operations Res.* 13 (1965) 517–546.
[2] N. Christofides, The shortest hamiltonian chain of a graph, *SIAM J. Appl. Math.* 19 (1970) 689–696.
[3] E.W. Dijkstra, A note on two problems in connection with graphs, *Num. Math.* 1 (1959) 269–271.
[3a] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, NJ, 1962).

[4] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming* (John Wiley, New York, 1972).

[5] A.M. Geoffrion, Integer programming by implicit enumeration and Balas' method, *SIAM Rev.* 9 (1967) 178–190.

[6] F. Glover, A multiphase-dual algorithm for the zero-one integer programming problem, *Operations Res.* 13 (1965) 879–919.

[7] F. Glover, D. Klingman and D. Karney, The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems, *Transportation Sci.* 6 (1972) 171–181.

[8] K.H. Hansen and J. Krarup, Improvements of the Held–Karp algorithm for the symmetric traveling-salesman problem, *Math. Programming* 7 (1974) 87–96.

[9] M. Held and R.M. Karp, A dynamic programming approach to sequencing problems, *SIAM* 10 (1962) 196–210.

[10] M. Held and R.M. Karp, The traveling-salesman problem and minimum spanning trees, *Operations Res.* 18 (1970) 1138–1162.

[11] M. Held and R. Karp, The traveling salesman problem and minimum spanning trees: II, *Math. Programming* 1 (1971) 6–25.

[12] M. Held, P. Wolfe and H.P. Crowder, Validation of subgradient optimization, *Math. Programming* 6 (1974) 62–88.

[13] E. Johnson, Networks and basic solutions, *Operations Res.* 14 (1966) 619–623.

[14] R.L. Karg and G.L. Thompson, A heuristic approach to solving traveling salesman problems, *Management Sci.* 10 (1964) 225–248.

[15] J. Kershenbaum and R. Van Slyke, Computing minimum trees, *Proc. ACM Annual Conference* (1972) 518–527.

[16] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.* 7 (1956) 48–50.

[17] J.D.C. Little et al., An algorithm for the traveling salesman problem, *Operations Res.* 11 (1963) 972–989.

[18] A.K. Obruca, Spanning tree manipulation and the traveling salesman problem, *Computer J.* 10 (1968) 374–377.

[19] R.C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* 36 (1957) 1389–1401.

[20] T.C. Raymond, Heuristic algorithm for the traveling-salesman problem, *IBM J. Res. Dev.* 13 (1969) 400–407.

[21] P. Rosenstiehl, L'arbre minimum d'un graphe, in: P. Rosenstiehl, ed., *Theory of Graphs* (Gordon and Breach, New York, 1967).

[22] V. Srinivasan and G.L. Thompson, Accelerated algorithms for labeling and relabeling of trees, with applications to distribution problems, *J. Assoc. Computing Machinery* 19 (1972) 712–726.

[23] G.L. Thompson, The stopped simplex method: I. Basic theory for mixed integer programming, integer programming, *Revue Francaise De Recherche Operationelle* 8 (1964) 159–182.

[24] G.L. Thompson, Pivotal operations for solving optimal graph problems, working paper.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 495–506

COMPUTATIONAL PERFORMANCE OF THREE SUBTOUR ELIMINATION ALGORITHMS FOR SOLVING ASYMMETRIC TRAVELING SALESMAN PROBLEMS*

T.H.C. SMITH

Department of Statistics, Rand Afrikaans University, Johannesburg, R.S.A.

V. SRINIVASAN

Graduate School of Business, Stanford University, Stanford, CA 94305, U.S.A.

G.L. THOMPSON

Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

In this paper we develop and computationally test three implicit enumeration algorithms for solving the *asymmetric* traveling salesman problem. All three algorithms use the assignment problem relaxation of the traveling salesman problem with subtour elimination similar to the previous approaches by Eastman, Shapiro and Bellmore and Malone. The present algorithms, however, differ from the previous approaches in two important respects:

(i) lower bounds on the objective function for the descendants of a node in the implicit enumeration tree are computed *without altering* the assignment solution corresponding to the parent node — this is accomplished using a result based on "cost operators",

(ii) a LIFO (*Last In, First Out*) depth first branching strategy is used which considerably reduces the storage requirements for the implicit enumeration approach. The three algorithms differ from each other in the details of implementing the implicit enumeration approach and in terms of the type of constraint used for eliminating subtours. Computational experience with randomly generated test problems indicates that the present algorithms are more efficient and can solve larger problems compared to (i) previous subtour elimination algorithms and (ii) the 1-arborescence approach of Held and Karp (as implemented by T.H.C. Smith) for the asymmetric traveling salesman problem. Computational experience is reported for up to 180 node problems with costs (distances) in the interval (1, 1000) and up to 200 node problems with bivalent costs.

1. Introduction

Excluding the algorithms of this paper, the state-of-the-art algorithms for the asymmetric traveling salesman problem appears to be that of [11] and more recently [1], both of which use the linear assignment problem as a relaxation (with subtour elimination) in a branch-and-bound algorithm. In the case of the *symmetric*

* This report was prepared as part of the activities of the Management Science Research Group, Carnegie-Mellon University, under Contract N00014-75-C-0621, NR 047-048 with the U.S. Office of Naval Research.

A considerably more detailed version of this paper is available (Management Sciences Research Report No. 369), and can be obtained by writing to the third author.

traveling salesman problem these algorithms as well as another interesting algorithm of Bellmore and Malone [1] based on the 2-matching relaxation of the symmetric traveling salesman problem are completely dominated in efficiency by the branch-and-bound algorithm of Held and Karp [10] (further improved in [8]) based on a 1-tree relaxation of the traveling salesman problem. In [13] an implicit enumeration algorithm using a LIFO (Last In First Out) depth first branching strategy based on Held and Karp's 1-tree relaxation was introduced and extensive computational experience indicates that algorithm to be even more efficient than the previous Held–Karp algorithms.

In [17] Srinivasan and Thompson showed how *weak lower bounds* can be computed for the subproblems formed in the Eastman–Shapiro branch-and-bound algorithm [5, 11]. The weak lower bounds are determined by the use of cell cost operators [14, 15] which evaluate the effects on the optimal value of the objective function of parametrically increasing the cost associated with a cell of the assignment problem tableau. Since these bounds are easily computable, it was suggested in [17] that the use of these bounds instead of the bounds obtained by resolving or post optimizing the assignment problem for each subproblem, would speed up the Eastman–Shapiro algorithm considerably. In this paper we propose and implement a straightforward LIFO implicit enumeration version of the Eastman–Shapiro algorithm as well as two improved LIFO implicit enumeration algorithms for the *asymmetric* traveling salesman problem. In all three of these algorithms the weak lower bounds of [17] are used to guide the tree search. The use of weak lower bounds in the branch-and-bound subtour elimination approach is explained with an example in [17].

We present computational experience with the new algorithms on problems of up to 200 nodes. The computational results indicate that the proposed algorithms are more efficient than (i) the previous subtour elimination branch-and-bound algorithms and (ii) a LIFO implicit enumeration algorithm based on the 1-arborescence relaxation of the asymmetric traveling salesman problem suggested by Held and Karp in [9], recently proposed and tested computationally in [12].

2. Subtour elimination using cost operators

Subtour elimination schemes have been proposed by Dantzig, et al. [3, 4], Eastman [5], Shapiro [11], and Bellmore and Malone [1]. The latter four authors use, as we do, the Assignment Problem (AP) relaxation of the traveling salesman problem (TSP) and then eliminate subtours of the resulting AP by driving the costs of the cells in the assignment problem away from their true costs to very large positive or very large negative numbers.

The way we change the costs of the assignment problem is (following [17]) to use the operator theory of parametric programming of Srinivasan and Thompson [14, 15]. To describe these let δ be a nonnegative number and (p, q) a given cell in the assignment cost matrix $C = [c_{ij}]$. A *positive (negative) cell cost operator* δC^+_{pq} (δC^-_{pq}) transforms the optimum solution of the original AP into an optimum solution of the problem AP$^+$(AP$^-$) with all data the same, except

$$c'_{pq} = c_{pq} + \delta; \ (c'_{pq} = c_{pq} - \delta).$$

The details of how to apply these operators are given in [14, 15] for the general case of capacitated transportation problems and in [17] for the special case of assignment problems. Specifically we note that μ^+_{pq} (μ^-_{pq}) denotes the *maximum extent* to which the operator δC^+_{pq}(δC^-_{pq}) can be applied without needing a primal basis change.

Denoting by Z the optimum objective function value for the AP, the quantity $(Z + \mu^+_{pq})$ is a lower bound (called a *weak lower bound* in [17]) on the objective function value of the optimal AP-solution for the subproblem formed by fixing (p, q) out. The quantity μ^+_{pq} can therefore be considered as a *penalty* (see [7]) for fixing (p, q) out. The important thing to note is that the penalty μ^+_{pq} can be computed from an assignment solution without changing it any way. Consequently, the penalties for the descendants of a node in the implicit enumeration approach can be efficiently computed without altering the assignment solution for the parent node.

In the subtour elimination algorithms to be presented next, it becomes necessary to "fix out" a basic cell (p, q), i.e., to exclude the assignment (p, q). This can be accomplished by applying the operator MC^+_{pq}, where M is a large positive number. Similarly a cell (p, q) that was previously fixed out can be "freed", i.e., its cost restored to its true value, by applying the negative cell cost operator. A cell can likewise be "fixed in" by applying MC^-_{pq}.

3. New LIFO implicit enumeration algorithms

The first algorithm (called TSP1) uses the Eastman–Shapiro subtour elimination constraints with the modification suggested by Bellmore and Malone [1, p. 304] and is a straightforward adaptation to the TSP of the implicit enumeration algorithm for the zero-one integer programming problem. We first give a stepwise description of algorithm TSP1:

Step 0. Initialize the node counter to zero and solve the AP. Initialize $ZB = M$ (ZB is the current upper bound on the optimal tour cost) and go to Step 1.

Step 1. Increase the node counter. If the current AP-solution corresponds to a tour, update ZB and go to Step 4. Otherwise find a shortest subtour and determine a penalty μ^+_{pq} for each edge in this subtour (if the edge has been fixed in, take $\mu^+_{pq} = M$, a large positive number, otherwise compute μ^+_{pq}). Let (p, q) be any edge in this subtour with smallest penalty μ^+_{pq}. If $Z + \mu^+_{pq} \geq ZB$, go to Step 4 (none of the edges in the subtour can be fixed out without Z exceeding ZB). Otherwise go to Step 2.

Step 2. Fix (p, q) out. If in the process of fixing out, $Z + p' \geq ZB$, go to Step 3. Otherwise, after fixing (p, q) out, push (p, q) on to the stack of fixed edges and go to Step 1.

Step 3. Free (p, q). If (q, p) is currently fixed in, go to Step 4. Otherwise fix (p, q) in, push (p, q) on to the stack of fixed edges and go to Step 1.

Step 4. If the stack of fixed edges is empty, go to Step 6. If the edge (p, q) on top of the stack has been fixed out in Step 2, go to Step 3. Otherwise, go to Step 5.

Step 5. Pop a fixed edge from the stack and free it (if it is a fixed in edge, restore the value of the corresponding assignment variable to one). Go to Step 4.

Step 6. Stop. The tour corresponding to the current value of ZB is the optimal tour.

In Step 1 of TSP1 we select the edge (p, q) to be fixed out as the edge in a shortest subtour with the smallest penalty. Selecting a shortest subtour certainly minimizes the number of penalty calculations while the heuristic of selecting the edge with the smallest penalty is intuitively appealing (but not necessarily the best choice). We tested this heuristic against that of selecting the edge with (i) the largest penalty among edges in the subtour (excluding fixed in edges) and (ii) the largest associated cost, on randomly generated asymmetric TSP's. The smallest penalty choice heuristic turned out to be three times as effective than (i) and (ii) on the average, although it did not do uniformly better on all test problems.

Every pass through Step 1 of algorithm TSP1 requires the search for a shortest subtour and once an edge (p, q) in this subtour is selected, the subtour is discarded. Later, when backtracking, we fix (p, q) in during Step 3 and go to Step 1 and again find a shortest subtour. This subtour is very likely to be the same one we discarded earlier and hence there is a waste of effort. An improvement of the algorithm TSP1 is therefore to save the shortest subtours found in Step 1 and utilize this information in later stages of computation. We found the storage requirements to do this were not excessive, so that this idea was incorporated into the next algorithm.

The second algorithm, called TSP2, effectively partitions a subproblem into mutually exclusive subproblems as in the scheme of Bellmore and Malone [1, p. 304] except that the edges in the subtour to be eliminated are considered in order of increasing penalties instead of the order in which they appear in the subtour. Whereas the search tree generated by algorithm TSP1 has the property that every nonterminal node has exactly two descendants, the nonterminal nodes of the search tree generated by algorithm TSP2 in general have more than two descendants. We now give a stepwise description of Algorithm TSP2. In the description we make use of the pointer S which points to the location where the Sth subtour is stored (i.e. at any time during the computation S also gives the level in the search tree of the current node).

Step 0. Same as in algorithm TSP1. In addition, set $S = 0$.

Step 1. Increase the node counter. If the current AP-solution corresponds to a tour, update ZB and go to Step 4. Otherwise increase S, find and store a shortest subtour as the Sth subtour (together with a penalty for each edge in the subtour, computed as in Step 1 of algorithm TSP1). Let (p, q) be any edge in this subtour with smallest penalty μ^+. If $Z + \mu^+ \geq ZB$, decrease S and go to Step 4 (none of the edges in the subtour can be fixed out without Z exceeding ZB). Otherwise go to Step 2.

Step 2. Same as in algorithm TSP1.

Step 3. Free (p, q). If all edges of the Sth subtour have been considered in Step 2, decrease S and go to Step 4. Otherwise determine the smallest penalty μ^+ stored with an edge (e, f) in the Sth subtour which has not yet been considered in Step 2. If $Z + \mu^+ < ZB$, fix (p, q) in, push (p, q) on to the stack of fixed edges, set $(p, q) = (e, f)$ and go to Step 2. Otherwise decrease S and go to Step 4.

Step 4. Same as in algorithm TSP1.

Step 5. Same as in algorithm TSP1.

Step 6. Same as in algorithm TSP1

The third algorithm, called algorithm TSP3, effectively partitions a subproblem into mutually exclusive subproblems as in the scheme of Garfinkel [6]. A stepwise description of the algorithm follows.

Step 0. Same as in algorithm TSP2

Step 1. Increase the node counter. If the current AP-solution corresponds to a tour, update ZB and go to Step 6. Otherwise increase S and store a shortest subtour as the Sth subtour (together with a penalty for each edge in the subtour, computed as in Step 2 of algorithm TSP1). Let (p, q) be the edge in this subtour with smallest penalty μ^+. If $Z + \mu^+ \geq ZB$, go to Step 5. Otherwise go to Step 2.

Step 2. Fix out all edges (p, k) with k a node in the Sth subtour. If in the process of fixing out, $Z + \mu^+ \geq ZB$, go to Step 3. Otherwise, when all these edges have been fixed out, go to Step 1.

Step 3. Free all fixed out (or partially fixed out) edges (p, k) with k a node in the Sth subtour. If all edges in the Sth subtour have been considered in Step 2, go to Step 4. Otherwise determine the smallest penalty μ^+ stored with an edge (e, f) in the Sth subtour which has not yet been considered in Step 2. If $Z + \mu^+ < ZB$, fix out all edges (p, k) with k not a node in the Sth subtour, let $p = e$ and go to Step 2. Otherwise go to Step 4.

Step 4. Free all edges fixed out for the Sth subtour and go to Step 5.

Step 5. Decrease S. If $S = 0$, go to Step 7. Otherwise go to Step 6.

Step 6. Let (p, k) be the last edge fixed out. Go to Step 3.

Step 7. Stop. The tour corresponding to the current value of ZB is the optimal tour.

Note that the fixing out of edges in step 3 is completely optional and not required for the convergence of the algorithm. If these edges are fixed out, the subproblems formed from a given subproblem do not have any tours in common (see [6]). Most of these edges will be nonbasic so that the fixing out process involves mostly cost changes. Only a few basis exchanges are needed for any edges that may be basic. However, there remains the flexibility of fixing out only selected edges (for example, only non-basic edges) or not fixing out of any of these edges.

4. Computational experience

Our major computational experience with the proposed algorithms is based on a sample of 80 randomly generated asymmetric traveling salesman problems with edge costs drawn from a discrete uniform distribution over the interval (1,1000). The problem size n varies from 30 to 180 nodes in a stepsize of 10 and five problems of each size were generated. All algorithms were coded in FORTRAN V and were run using only the core memory (approximately 52,000 words) on the UNIVAC 1108 computer.

We report here only our computational experience with algorithms TSP2 and TSP3 on these problems since algorithm TSP1 generally performed worse than either of these algorithms, as could be expected a priori.

In Table 2 we report, for each problem size, the average runtimes (in seconds) for solving the initial assignment problem using the 1971 transportation code of

Table 2.

Summary of computational performance of algorithms TSP2 and TSP3

Problem size n	Average time to obtain assignment solution	Average runtime (including the solution of the AP) TSP2	Average runtime (including the solution of the AP) TSP3	Algorithm TSP2: Average runtime estimated by regression	Algorithm TSP2: Average time to obtain first tour	Algorithm TSP2: Average quality of first tour (% from optimum)
30	0.2	0.9	1.0	0.8	0.3	3.7
40	0.4	2.9	2.8	1.9	0.5	4.0
50	0.3	1.7	3.4	3.9	0.6	6.8
60	0.9	9.3	11.4	6.9	1.5	4.1
70	[illegible]	6.7	11.8	11.2	1.3	0.5
80	1.5	13.8	[illegible]	17.3	2.3	1.0
90	1.9	42.0	56.5	25.2	3.6	7.7
100	2.1	53.0	59.6	35.3	5.2	3.8
110	2.8	77.3	—	47.6	3.7	1.9
120	3.5	62.9	—	62.8	5.7	1.5
130	4.0	[illegible]		80.9	8.3	2.0
140	5.6	165.3	—	107.4	17.9	4.7
150	6.1	65.3	—	127.6	9.0	1.1
160	7.0	[illegible]		156.6	10.9	[illegible]
170	8.0	169.8	—	180.9	13.3	1.2
180	8.9	441.4	—	227.7	23.0	2.1

Note. (1) All averages are computed over 5 problems each.
(2) All computational times are in seconds on the UNIVAC 1108.

Srinivasan and Thompson [16] as well as the average runtime (in seconds including the solution of the AP) for algorithms TSP2 and TSP3. From the results for $n \leq 100$, it is clear that algorithm TSP2 is more efficient than TSP3. For this reason, only algorithm TSP2 was tested on problems with $n > 100$. We determined that the function $t(n) = 1.55 \times 10^{[illegible]} \times n^{[illegible]}$ fits the data with a coefficient of determination (R^2) of 0.927. The estimated runtimes obtained from this function are also given in Table 1.

It has been suggested that implicit enumeration or branch-and-bound algorithms can be used as approximate algorithms by terminating them as soon as a first solution is obtained. In order to judge the merit of doing so with algorithm TSP2, we also report in Table 1 the average runtime (in seconds) to obtain the first tour as well as the quality of the first tour (expressed as the difference between the first tour cost and the optimal tour cost as a percentage of the latter). Note that for all n the first tour is, on an average, within 5% of the optimum and usually much closer.

We mentioned above that the fixing out of edges in step 3 of algorithm TSP3 is not necessary for the convergence of the algorithm. Algorithm TSP3 was temporarily modified by eliminating the fixing out of these edges but average runtimes increased significantly (the average runtimes for the 70 and 80 node problems were respectively 24.3 and 25.5 seconds). Hence it must be concluded that the partitioning scheme introduced by Garfinkel [6] has a practical advantage over the original branching scheme of Bellmore and Malone [1].

The largest asymmetric TSP's solved so far appears to be two 80-node problems solved by Bellmore and Malone [1] in an average time of 165.4 seconds on an IBM 360/65. Despite the fact that the IBM 360/65 is somewhat slower (takes about 30 to 50% longer time) compared to the UNIVAC 1108, the average time of 13.8 seconds for TSP2 on the UNIVAC 1108, is still considerably faster than the Bellmore–Malone [1] computational times. Svestka and Huckfeldt [18] solved 60 node problems on a UNIVAC 1108 in an average time of 80 seconds (vs. 9.3 seconds for algorithm TSP2 on a UNIVAC 1108). They also estimated the average runtime for a 100 node problem as 27 minutes on the UNIVAC 1108 which is considerably higher than that required for TSP2.

The computational performance of algorithm TSP2 was also compared with the LIFO implicit enumeration algorithm in [12] for the asymmetric traveling salesman problem using Held and Karp's 1-arborescence relaxation. The 1-arborescence approach reported in [12] took, on the average, about 7.4 and 87.7 seconds on the UNIVAC 1108 for $n = 50$ and 60 respectively. Comparison of these numbers with the results in Table 1 again reveals that TSP2 is computationally more efficient. For the *symmetric* TSP, however, algorithm TSP2 is completely dominated by a LIFO implicit enumeration approach with the Held–Karp 1-tree relaxation. See [13] for details.

A more detailed breakdown of the computational results are presented in Table 2 (for TSP2 and TSP3 for $n \leq 100$) and in Table 3 (for TSP2 for $n > 100$). The column headings of Tables 2 and 3 have the following interpretations:

Table 2.

Computational characteristics of algorithms TSP2 and TSP3 for n ≤ 100.

Problem	Gap	Pivots		Nodes		Penalties		Maximum subtrees stored		Runtime (secs)	
		TSP2	TSP3	TSP2	TSP3	TSP2	TSP3	TSP2	TSP3	TSP2	TSP3
P30-1	2.48	187	186	11	12	173	177	4	4	0.7	0.7
P30-2	3.29	174	194	6	6	[illegible]	342	3	4	0.5	0.6
P30-3	1.41	410	344	34	34	468	386	10	7	1.6	1.4
P30-4	1.62	175	464	14	14	[illegible]	409	4	1	0.7	1.3
P30-5	4.06	250	251	17	17	280	290	7	6	1.0	1.0
P40-1	2.52	127	137	5	5	42	52	3	3	0.4	0.5
P40-2	2.94	352	657	16	16	381	686	5	6	1.9	2.8
P40-3	4.64	1278	1136	58	51	1874	1459	10	10	7.7	6.9
P40-4	1.13	143	177	7	7	77	115	5	3	0.6	0.8
P40-5	0.24	572	514	44	35	786	627	10	6	3.7	3.0
P50-1	0.20	144	154	2	2	6	6	1	1	0.4	0.5
P50-2	0.39	171	173	3	3	19	21	2	2	0.5	0.4
P50-3	1.65	541	1307	31	51	513	1768	3	[illegible]	3.9	10.9
P50-4	2.56	257	300	7	7	192	736	4	4	1.5	1.6
P50-5	3.28	340	877	11	11	570	918	5	5	2.3	4.2
P60-1	1.22	524	2935	13	112	428	4176	7	10	4.1	30.4
P60-2	2.42	559	1099	24	22	605	1147	5	6	4.9	7.0
P60-3	0.77	1611	1029	55	47	1811	1029	12	7	12.5	9.2
P60-4	1.64	2164	1260	92	32	3279	1349	20	7	24.2	19.1
P60-5	0.53	266	268	3	3	27	29	3	3	1.0	0.9
P70-1	1.16	440	507	8	5	780	572	4	4	7.4	4.1
P70-2	1.52	1865	2676	94	103	2650	3715	14	13	22.9	34.6
P70-3	2.29	309	310	5	5	70	51	2	2	1.2	1.2
P70-4	1.79	677	603	21	21	521	378	7	7	6.4	7.7
P70-5	3.05	1006	1347	34	34	988	1375	6	6	8.9	11.5
P80-1	3.36	1065	1078	36	35	1154	1210	10	9	12.5	13.2
P80-2	0.47	819	885	25	25	927	965	8	7	9.2	10.1
P80-3	1.34	1759	2348	40	47	1934	2636	9	8	21.4	26.4
P80-4	1.29	1357	1397	25	27	1543	1658	6	6	15.6	17.6
P80-5	1.37	537	944	39	37	596	953	8	8	9.6	11.7
P90-1	0.04	513	570	3	3	165	222	2	2	4.4	4.5
P90-2	0.87	842	831	7	5	353	343	3	3	6.3	5.9
P90-3	1.17	5868	7391	226	217	9259	10650	44	39	108.1	120.8
P90-4	0.99	1822	4282	47	107	1955	5008	7	10	24.6	67.8
P90-5	0.81	2596	4867	140	139	4990	6353	17	17	46.7	56.0
P100-1	1.84	3069	804	90	8	4794	756	17	4	61.4	5.9
P100-2	0.72	1387	1741	35	36	1530	1914	9	10	25.5	29.8
P100-3	0.54	4341	6812	244	348	6157	12649	17	29	94.3	[illegible]
P100-4	0.94	602	628	8	8	193	226	5	5	4.8	4.8
P100-5	0.85	3770	3990	106	66	6255	4237	24	12	81.1	73.0

Table 5.

Computational characteristics of algorithm JSP2 for n ≥ 100.

Problem	Gap	Pivots	Nodes	Penalties	Maximum subtours stored	Runtime (secs.)
P110-1	0.98	2958	44	3605	13	52.0
P110-2	0.62	1225	25	1353	6	18.9
P110-3	0.36	1141	22	699	7	14.7
P110-4	0.83	1525	14	1152	5	25.0
P110-5	0.09	719	2	9	1	2.9
P120-1	0.89	2754	74	3237	11	62.7
P120-2	0.45	2044	61	2495	13	46.7
P120-3	0.31	1526	31	1431	5	25.9
P120-4	1.07	1311	20	939	7	16.8
P120-5	1.12	5845	129	9435	14	159.5
P130-1	0.20	4251	104	11910	17	215.4
P130-2	0.06	1985	44	1804	8	40.0
P130-3	2.16	5968	139	5063	12	152.7
P130-4	0.12	3107	77	4264	13	85.2
P130-5	0.49	2557	40	2615	9	56.1
P140-1	0.65	1757	26	1067	8	27.4
P140-2	0.54	1568	17	895	6	24.4
P140-3	1.49	11109	319	19591	49	407.1
P140-4	1.22	9772	236	13694	37	307.6
P140-5	0.06	2274	52	2740	10	59.3
P150-1	0.81	1491	20	769	5	23.5
P150-2	0.64	4139	84	4912	16	126.4
P150-3	0.49	1597	14	680	6	21.9
P150-4	1.29	2915	61	2675	10	71.1
P150-5	0.86	2788	73	3151	15	78.5
P160-1	0.11	3729	79	4923	10	120.5
P160-2	0.40	3683	66	4056	12	105.0
P160-3	0.62	3563	54	3314	13	92.5
P160-4	0.29	3250	29	4363	16	105.8
P160-5	0.50	3615	71	4422	11	118.9
P170-1	0.06	4133	77	4393	10	125.9
P170-2	0.40	3049	40	2854	7	95.9
P170-3	0.68	4311	66	4119	11	135.1
P170-4	0.55	4196	110	6032	14	175.5
P170-5	0.12	6056	195	12577	17	330.4
P180-1	1.37	12525	221	19071	24	574.2
P180-2	0.56	7115	189	10614	22	304.4
P180-3	0.31	13049	299	21030	27	609.2
P180-4	2.50	9292	199	14930	24	430.1
P180-5	0.38	7701	175	9166	20	289.1

Problem:	The ith problem of size n is identified as n–i.
Gap:	The difference between the optimal assignment cost and the optimal tour cost as a percentage of the optimal tour cost.
Pivots:	The total number of basis exchanges.
Nodes:	The number of nodes in the search tree generated (i.e. the final value of the node counter used in the algorithm descriptions).
Penalties:	The total number of times that μ^+ or μ^- were computed (either as a penalty or in the process of fixing out or freeing a cell).
Maximum Subtours Stored:	The maximum number of subtours stored simultaneously (i.e. the maximum depth of a node in the search tree generated).
Runtime:	The total runtime in seconds on the UNIVAC 1108 including the time for solving the AP but excluding time for problem generation.

From Tables 2 and 3 we find that the maximum number of subtours that had to be stored for a problem of size n was always less than $n/3$ except for a 90 node problem which had 34 maximum subtours and a 140 node problem which had 49 maximum subtours. Thus allowing for a storage of a maximum of about $n/2$ subtours should suffice almost always.

In [2] Christofides considers asymmetric traveling salesman problems with bivalent costs — i.e. each cost c_{ij}, $i \neq j$, can have only one of two values. He conjectured that this type of problem would be "difficult" for methods based on subtour elimination and hence proposed and tested a graph-theoretical algorithm for these special traveling salesman problems. In the testing of his algorithm (on a CDC 6600) he made use of six problems ranging in size from 50 to 500 nodes. These problems were randomly generated with an average of four costs per row being zero and all nonzero costs having the value one (except for diagonal elements which were M as usual).

For each of the problem sizes 50, 100, 150 and 200 we generated five problems (i.e. twenty problems altogether) with zero–one cost matrices (except for diagonal elements) which have the same type of distribution of zeros as Christofides' problems. We solved the problems with fewer than 200 nodes with both algorithms TSP1 and TSP2 and the five 200 node problems with algorithm TSP1 only (because of core limitations on the UNIVAC 1108 we are limited to 200-node problems for algorithm TSP1 and 180-node problems for algorithm TSP2).

The average runtimes (in seconds) for each problem size are reported in Table 4. The last column of Table 4 contains the CDC 6600 runtime (in seconds) obtained by Christofides on a problem of the given size. Since the CDC 6600 is generally regarded as faster (takes about 10–50% less time) compared to the UNIVAC 1108, algorithms TSP1 and TSP2 can be regarded as more efficient than the algorithm in [2]. An interesting observation was that for all the problems of this type which were solved, the optimal assignment cost equalled the optimal tour cost (i.e., an optimal AP solution is also optimal to the TSP).

Table 4

Computational comparisons for bivalent cost asymmetric traveling salesman problems

Problem size n	Average runtime* (UNIVAC 1108 secs.) TSP1	TSP2	Christofides' [2] runtime (CDC 6600 secs.)
50	0.5	0.6	9.5
100	2.4	2.5	15.5
150	5.4	5.4	–
200	6.4	—	12.8

* Average based on 5 problems each.

5. Conclusion

We have proposed new algorithms for the asymmetric traveling salesman problem and presented extensive computational experience with these algorithms. The results show that our algorithms are:

(i) more efficient than earlier algorithms and

(ii) capable of solving problems of more than twice the size previously solved.

In view of the ongoing research on transportation algorithms and the improvements in computer performance, it is likely that the proposed algorithms will be able to solve much larger traveling salesman problems in the near future.

References

[1] M. Bellmore and J.C. Malone, Pathology of traveling salesman subtour-elimination algorithms, *Operations Res.* 19 (1971) 278–307.

[2] N. Christofides, Large scheduling problems with bivalent costs, *Computer J.* 16 (1973) 262–264.

[3] G.B. Dantzig, D.R. Fulkerson and S.M. Johnson, Solution of a large scale traveling salesman problem, *Operations Res.* 2 (1954) 393–410.

[4] G.B. Dantzig, D.R. Fulkerson and S.M. Johnson, On a linear programming, combinatorial approach to the traveling salesman problem, *Operations Res.* 7 (1959) 58–66.

[5] W.L. Eastman, Linear programming with pattern constraints, Unpublished Ph.D. Dissertation, Harvard University (1958).

[6] R.S. Garfinkel, On partitioning the feasible set in a branch-and-bound algorithm for the asymmetric traveling salesman problem, *Operations Res.* 21 (1973) 340–343.

[7] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming* (John Wiley, New York, 1972).

[8] K.H. Hansen and J. Krarup, Improvements of the Held–Karp algorithm for the symmetric traveling salesman problem, *Math. Programming* 7 (1974) 87–96.

[9] M. Held and R.M. Karp, The traveling salesman problem and minimum spanning trees, *Operations Res.* 18 (1970) 1138–1162.

[10] M. Held and R.M. Karp, The traveling salesman problem and minimum spanning trees: Part II, *Math. Programming* 1 (1971) 6–25.

[11] D.M. Shapiro, Algorithms for the solution of the optimal cost and bottleneck traveling salesman problems, unpublished Sc.D. Thesis, Washington University, St. Louis, (1966).

[12] T.H.C. Smith, A LIFO implicit enumeration algorithm for the asymmetric traveling salesman problem using a 1-arborescence relaxation, Management Science Research Report No. 380, Graduate School of Industrial Administration, Carnegie-Mellon University (1975).
[13] T.H.C. Smith and G.L. Thompson, A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation, *Ann. Discrete Math.* 1 (1977) 479–493.
[14] V. Srinivasan and G.L. Thompson, An operator theory of parametric programming for the transportation problem-I, *Naval Res. Logistics Quarterly* 19 (1972) 205–225.
[15] V. Srinivasan and G.L. Thompson, An operator theory of parametric programming for the transportation problem-II, *Naval Res. Logistics Quarterly* 19 (1972) 227–252.
[16] V. Srinivasan and G.L. Thompson, Benefit-cost analysis of coding techniques for the primal transportation algorithm, *J. Assoc. Computing Machinery* 20 (1973) 194–213.
[17] V. Srinivasan and G.L. Thompson, Solving scheduling problems by applying cost operators to assignment models, in: S.E. Elmaghraby (Ed.), Symp. *Theory of Scheduling and its Applications*, (Springer, Berlin, 1973) 399–425.
[18] J. Svestka and V. Huckfeldt, Computational experience with an M-salesman traveling salesman algorithm, *Management Sci.* 19 (1973) 790–799.

Annals of Discrete Mathematics 1 (1977) 507-515

ON ANTIBLOCKING SETS AND POLYHEDRA

Jørgen TIND

Institut for Operationsanalyse, Aarhus Universitet, c/o Matematisk Institut Ny Munkegade, 8000 Aarhus C, Denmark.

This paper first gives an economic interpretation of the duality correspondence for antiblocking sets and polyhedra, which at least in the polyhedral case play an important role in the study of certain integer programming problems, e.g. covering problems. We then discuss, in view of the duality correspondence, how bounds for such problems may be obtained by relatively simple network flow methods.

1. Introduction

This paper gives an economic interpretation of the duality relationship for a pair of antiblocking sets/polyhedra. The interpretation is similar to the one given by A.C. Williams for conjugate, convex functions [8]. But here in the antiblocking framework they are replaced by antiblocking, concave functions, a concept related to polar functions [5]. As in [8] we also consider the relationship between a manufacturer and a contractor, who wants to compute a minimal compensation for taking over the production activities from the manufacturer. For that purpose the contractor quotes unit prices on the activities. The manufacturer's objective is here to minimize his average cost per unit produced. By the duality relationship for antiblocking sets it is then shown that the selected price mechanism operates in a natural way such that it makes no difference for the manufacturer, if he produces by himself or not.

The concept of antiblocking sets is a generalization of antiblocking polyhedra, which have been introduced by Fulkerson [1] and which have been shown to be an excellent framework for consideration of many combinatorial problems. One of these problems is the covering problem. The last part of the paper is devoted to an idea for computation of bounds for such problems by means of chains or antichains in constructed networks. The idea has previously been used in [4] for the set partitioning problem.

In order to avoid lengthening this paper the relating blocking framework is not considered here, even though a similar discussion may be developed for this case, too.

2. Antiblocking sets

Let $B \subseteq \mathbf{R}^n$ be a closed, convex set, containing 0. The *polar set* B^* of B is defined as

$$B^* = \{x^* \in \mathbf{R}^n \mid x \cdot x^* \leq 1, \forall x \in B\}.$$

B^* is also a closed, convex set that contains 0.

Additionally we have that $B^{**} = B$, i.e. B is again the polar set of B^*. This is the Minkowski polarity correspondence (see e.g. [5, section 14]).

Let also $D \subseteq \mathbf{R}^n$ be a closed convex set containing 0. Define the *antiblocking set* $\bar{B} \subseteq \mathbf{R}^n$ of B with respect to D as follows:

$$\bar{B} = B^* \cap D.$$

In the following we will investigate conditions under which

$$\bar{\bar{B}} = B, \tag{2.1}$$

i.e., when B is the antiblocking set of $\bar{B}$ with respect to D. In that case B and $\bar{B}$ are called a *pair of antiblocking sets*.

It is seen that with $D = \mathbf{R}^n$ we are back in the Minkowski polarity. But in more general cases it is necessary to impose special conditions on B in order to show the polarity correspondence in (2.1).

Let cl C denote the closure of C and let conv C denote the convex hull of C, where $C \subseteq \mathbf{R}^n$. We then have the following theorem which gives a necessary and sufficient condition for equation (2.1) to be valid.

Theorem 2.1. *$\bar{\bar{B}} = B$, if and only if $B = \mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D$.*

Proof. $\bar{\bar{B}} = (\bar{B})^* \cap D = (B^* \cap D)^* \cap D$.

We have for polar sets in general that

$$(B \cap D)^* = \mathrm{cl}\,\mathrm{conv}\,(B^* \cup D^*)$$

(see [5, Corollary 16.5.2]). Hence with B replaced by B^* we get that

$$\bar{\bar{B}} = \mathrm{cl}(\mathrm{conv}(B^{**} \cup D^*)) \cap D = \mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D.$$

The next theorem gives another set of conditions that are necessary and sufficient for (2.1) to be valid. These conditions can especially be applied when B is described as the intersection of halfspaces.

Theorem 2.2.[1] *$\bar{\bar{B}} = B$, if and only if there exists a closed, convex set $C \subseteq \mathbf{R}^n$ containing 0 such that $B = C \cap D$ and such that $C^* \subseteq D$.*

[1] For polyhedra, Theorem 2.2 is a special case of joint work by Jørgen Aráoz, Jack Edmonds and Victor Griffin. Personal communication.

Proof. Let us first assume that $\bar{B} = B$, and let $C = \mathrm{cl}(\mathrm{conv}(B \cup D^*))$. Obviously, C is closed, convex and $0 \in C$. Theorem 2.1 implies that $B = C \cap D$. Additionally, as $C \supseteq D^*$, we have that $C^* \subseteq D^{**} = D$. This shows one direction of the theorem.

Now assume that we have a set C such that $B = C \cap D$ and $C^* \subseteq D$. [It is remarked that C here in general might be different from the previous set $\mathrm{cl}(\mathrm{conv}(B \cup D^*))$]. From theorem 2.1 it is now sufficient to show that

$$B = \mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D.$$

Since $B = B \cap D \subseteq \mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D$, it is enough to show the reverse inclusion:

$$B \supseteq \mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D. \tag{2.2}$$

By assumption $C^* \subseteq D$, which implies that $C \supseteq D^*$. Moreover, $C \supseteq B$. Since C is closed and convex, we obtain that $C \supseteq \mathrm{cl}(\mathrm{conv}(B \cup D^*))$. Hence $\mathrm{cl}(\mathrm{conv}(B \cup D^*)) \cap D \subseteq C \cap D = B$, where the last equation follows by assumption. This shows (2.2), and the theorem is proved.

The assumption $C^* \subseteq D$ in the theorem expresses in particular that all supporting hyperplanes for C have their normals contained in D. (The defining linear forms are normalised ($= 1$)).

If $D = \mathbf{R}^n_+ = \{x \in \mathbf{R}^n \mid x \geq 0\}$ and $C = \{x \in \mathbf{R}^n \mid Ax \leq 1\}$, where A is an $m \times n$ matrix of nonnegative elements and $1 = (1, \ldots, 1)$ with m elements, then the theorem can be applied on $B = C \cap D$. In this case B and $\bar{B}$ constitute a pair of antiblocking polyhedra [1].

Theorem 2.2 is an extension of a result in [6].

A similar discussion can also be made for blocking sets and polyhedra ([6] and [7]).

3. A geometrical illustration of antiblocking sets

The relation between B and its polar set B^* can be given in the following equivalent way.

$$B^* = \{x^* \in \mathbf{R}^n \mid (x, 1) \cdot (x^*, -1) \leq 0, \forall x \in B\},$$

where $(x, 1)$ and $(x^*, -1)$ are vectors in $\mathbf{R}^{n+1}$.

Hence, if we consider the space $\mathbf{R}^{n+1}$ and let B be placed in the hyperplane H^{+1}, where $H^{+1} = \{(x, 1) \mid x \in \mathbf{R}^n\}$, then B^* can be obtained as follows. Construct the cone P generated by B with vertex at $0 \in \mathbf{R}^{n+1}$ and its polar cone $P^* = \{y^* \in \mathbf{R}^{n+1} \mid y \cdot y^* \leq 0, \forall y \in P\}$. Where P^* intersects the hyperplane $H^{-1} = \{(x, -1) \in \mathbf{R}^{n+1} \mid x \in \mathbf{R}^n\}$ we get an image of B^*. $\bar{B}$ is now by definition obtained as the intersection of B^* and D.

Let us look at the situation where $D = \mathbf{R}^2_+ = \{x \in \mathbf{R}^2 \mid x \geq 0\}$. Fig. 1 gives now an

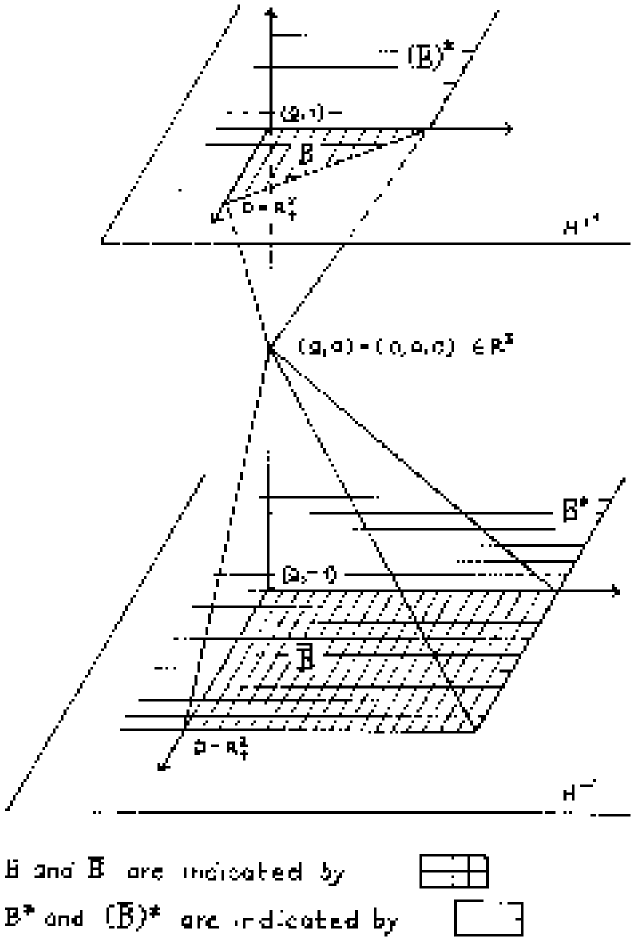


Fig. 1.

illustration of a set $B \subseteq \mathbb{R}^2_+$ and its antiblocking set $\bar{B} \subseteq \mathbb{R}^2$ in the situation where $\bar{\bar{B}} = B$. Here B and $\bar{B}$ are actually polyhedra.

4. An economic interpretation

Consider a concave, nonnegative closed function $f(x): \mathbb{R}^n_+ \to \mathbb{R}_+$.

Let $\mathrm{sub}_+ f \subseteq \mathbb{R}^{n+1}$ denote the nonnegative subgraph of $f(x)$, i.e.

$$\mathrm{sub}_+ f = \{(x, y) \in \mathbb{R}^{n+1} \mid x \geq 0,\ 0 \leq y \leq f(x)\}.$$

Since $f(x)$ is a nonnegative function, it is uniquely determined by $\mathrm{sub}_+ f$.

With the given specifications on $f(x)$ it follows that $\mathrm{sub}_+ f$ is a closed, convex set, containing 0.

Define the following function $f(x^*): \mathbb{R}^n_+ \to \mathbb{R}_+$.

$$\bar{f}(x^*) = \sup\{y^* \in R \mid -x \cdot x^* + y^* f(x) \leq 1, \ \forall x \geq 0\}.$$

Call $\bar{f}(x^*)$ *the antiblocking function* of f. $\bar{f}(x^*)$ becomes also nonnegative, and its nonnegative subgraph is given by

$$\text{sub}_+ \bar{f} = \{(x^*, y^*) \in \mathbf{R}^{n+1} \mid (x^*, y^*) \cdot (-x, y) \leq 1$$
$$\text{for all } (x, y) \in \text{sub}_+ f\} \cap \{(x^*, y^*) \in \mathbf{R}^{n+1} \mid (x^*, y^*) \geq 0\}.$$

This shows that $\bar{f}$ is concave and closed.

Let T denote the linear transformation $T: (x, y) \rightarrow (-x, y)$. If $D = \mathbf{R}^{n+1}_+$, it is seen that $\text{sub}_+ \bar{f}$ is obtained by the antiblocking relation with respect to D as follows.

$$\text{sub}_+ \bar{f} = \overline{T(\text{sub}_+ f)}.$$

Additionally it is assumed that $f(x)$ is a non-decreasing function in each component, which implies that all supporting hyperplanes for $T(\text{sub}_+ f)$ have their normals in D. Hence, it is obtained by the same reasoning as in the proof of theorem 2.1, that

$$\overline{T(\text{sub}_+ \bar{f})} = \text{sub}_+ f.$$

This shows that

$$\bar{\bar{f}} = f, \tag{4.1}$$

i.e., f is the antiblocking function of $\bar{f}$.

We will try to give an economic interpretation of this equation in the following.

Note that $\bar{f}(x^*)$ can be expressed alternatively as

$$\bar{f}(x^*) = \inf_{x \geq 0} \frac{x^* \cdot x + 1}{f(x)},$$

where $(x^* \cdot x + 1)/f(x) = \infty$, if $f(x) = 0$.

The polarity will not be disturbed by rescaling, i.e. by replacement of the number 1 by an arbitrary number $k > 0$, which means that

$$\bar{f}(x^*) = \inf_{x \geq 0} \frac{x \cdot x^* + k}{f(x)}.$$

Assume now that a manufacturer produces a product by means of n activities. Let the components of the vector $x \geq 0$ denote the activity level of each activity. With a given activity level he produces $f(x)$ units of the product. Assume additionally that the components of the vector $x^* \geq 0$ denote market prices that equal the cost for use or consumption of one unit of the corresponding activities. Hence $x \cdot x^*$ is a cost for production of $f(x)$ units of the product. In addition to this cost, which is linear in x, there is supposed to be a constant cost of size k. It is further supposed that the manufacturer's objective is to minimize the average cost per unit produced, i.e., the manufacturer wants to solve the following problem

$$\bar f(x^*) = \inf_{x \geq 0} \frac{x \cdot x^* - k}{f(x)}.$$

Hence the antiblocking function of f denotes the minimal average cost, given a price vector x^*.

Now the manufacturer also considers selling his activities at a given level $x \geq 0$ to a contractor, who in return should pay him with an amount of the finished product. For that purpose the contractor quotes a unit price $x^* \geq 0$ on each activity. Based on this price the manufacturer at least would demand an amount of the finished product that equals the estimated production costs, divided by the average cost per unit, i.e.

$$\frac{x \cdot x^* - k}{\bar f(x^*)}.$$

Hence, the contractor, seeing no reason to return more than that amount, will get the task to find a price $x^* \geq 0$ that solves the problem

$$\bar{\bar f}(x) = \inf_{x^* \geq 0} \frac{x \cdot x^* - k}{\bar f(x^*)}.$$

By (4.1) we get the reasonable result that with such a price the amount of finished product does not depend on whether he produces by himself or lets the contractor do it for him.

It is remarked that the idea of antiblocking functions is almost the same as the idea of polar functions in [5, section 15]. But again, the polarity is considered with respect to a given set, here $\mathbf{R}^n_+$. This has the effect that the prices x^* are nonnegative, and the function $f(x)$ is non-decreasing, which seems reasonable in the economic context above.

5. Bounds for set-covering problems

From the preceding discussion it is seen that the antiblocking relation itself is developed over the continuous space $\mathbf{R}^n$. But historically the concept came up through studies of discrete problems, especially certain integer programming problems [1].

In the following discussion one of these integer programming problems will be examined, in view of the duality for antiblocking polyhedra. An example will be given, which illustrates how one may obtain computationally simple bounds for the value of those problems. The idea for construction of these bounds has previously been developed and used in [4] for the set-partitioning problem, and the following material is highly related to this work.

Here we will look at the following set covering problem:

$$\begin{aligned} &\min \mathbf{1}\cdot x\\ &\qquad Ax \geq w \qquad\qquad (5.1)\\ &\qquad\; x \geq 0 \quad \text{and integer}, \end{aligned}$$

where A is an $m \times n$ matrix of zeros and ones, w is a nonnegative integer m-vector, and $\mathbf{1} = (1, \ldots, 1)$ with n elements. By removal of the integrality requirement we get:

$$\begin{aligned} &\min \mathbf{1}\cdot x\\ &\qquad Ax \geq w \qquad\qquad (5.2)\\ &\qquad\; x \geq 0, \end{aligned}$$

which, as usual, by standard LP gives a lower bound for the objective function in (5.1). But in some cases a bound can be obtained even simpler. For example, if the columns in A are incidence vectors for all maximal chains in an oriented network without cycles, then the problem can be solved by an algorithm of the network flow type. For instance, Connect all endpoints of the chains to a source and a sink, respectively. Place a lower bound of w_i (the ith component of w) on the ith node, and compute the minimal flow from s to t.

With $w = (1, \ldots, 1)$ this problem is a generalization of one part of the Dilworth Theorem. See for example [3]. The result is integer. This is seen directly, or in more general terms from the antiblocking theory this follows by the min-max equality, which here holds for A and $\bar{A}$. The columns in $\bar{A}$ are the incidence vectors of all maximal antichains in the same network, and the set $B = \{y \geq 0 \mid yA \leq 1\}$, (which is the dual constraint set of (5.2)) and the set $\bar{B} = \{y^* \geq 0 \mid y^*\bar{A} \leq 1\}$ constitute a pair of antiblocking polyhedra. See [1].

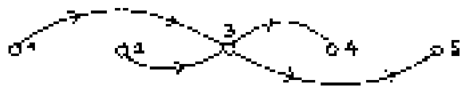
Now generally A is not the incidence matrix of all maximal chains in a network. But a network can be constructed in which A is the incidence column matrix of at least some chains. Then, by solving the covering problem over all chains, we receive a lower bound for (5.1). This lower bound is easy to compute, although in general it is weaker than the bound obtained by solving (5.2).

Consider the following example, where $w = (1, \ldots, 1)$, [4]:

$$\min \sum_{i=1}^{4} x_i$$

$$\begin{pmatrix} 1 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 0\\ 1 & 0 & 0 & 1\\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1\\ \cdot\\ \cdot\\ \cdot\\ x_4 \end{pmatrix} \geq \begin{pmatrix} 1\\ \cdot\\ \cdot\\ \cdot\\ 1 \end{pmatrix}. \qquad (5.3)$$

Let the nodes in the network be numbered corresponding to the row numbers of A. Then the network looks as follows,

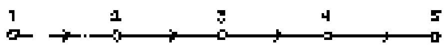


and the bound is 2 (the minimal number of chains that cover all nodes, which is equal to the maximal size of an antichain; Dilworth).

The result is generally dependent on the permutation of rows. For example, with the matrix:

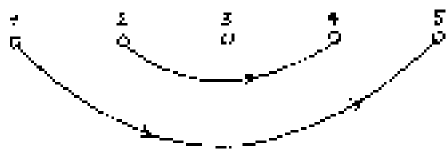
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix},$$

we get the network



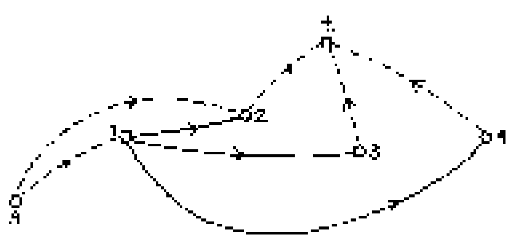
and the bound is equal to 1.

We can also construct a loopless oriented network, in which the matrix A corresponds to some of the antichains in the network. For the problem (5.3) the network may look like the following:



The minimal number of covering antichains, which is equal to the largest chain (the companion to the Dilworth theorem [1]) gives a lower bound. The result here is 2.

An *upper* bound for the set covering problem can be found in a similar way by network flow methods, where now the *rows* of the matrix are incidence vectors for chains (or antichains). For example, with chains we get the following network for the problem (5.3):



The numbers correspond to the columns. The endpoints of the chains are connected to a source s and a sink t, respectively. The problem is now to find a minimal number of nodes that block all s–t chains (which is equal to the maximal number of node independent chains from s to t; Menger's Theorem). Here the result is 2.

We believe that such bounds may be helpful in an algorithm for solution of set covering type problems, and an algorithm incorporating that feature is now under development.

Acknowledgement

I wish to thank J. Aráoz for his useful comments on an earlier version of this paper.

References

[1] D.R. Fulkerson, Anti-blocking polyhedra, J. Comb. Theory 12 (1972) 50–71.

[2] D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra, *Math. Programming* 1 (1971) 168–194.

[3] D.R. Fulkerson, Flow networks and combinatorial operations research, *Am. Math. Monthly* 73 (1966) 115–138.

[4] G.L. Nemhauser, L.E. Trotter, Jr. and R.M. Nauss, Set partitioning and chain decomposition, *Management Sci.* 20 (1974) 1413–1423.

[5] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).

[6] J. Tind, Blocking and antiblocking sets, Math. Programming 6 (1974) 157–166.

[7] J. Tind, Dual correspondences for blocking and antiblocking sets (1974), 10 pp., Institut for Operationsanalyse, University of Aarhus; or Report No. 7421-OR, Institut für Ökonometrie und Operations Research, University of Bonn.

[8] A.C. Williams, Nonlinear activity analysis, *Management Sci.* 17 (1970) 127–139.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 517–525

ON THE GENERALITY OF MULTI-TERMINAL FLOW THEORY

L.E. TROTTER, Jr.*

Department of Operations Research, College of Engineering, Cornell University, Ithaca, NY, U.S.A.

We consider the problem of determining maximal flows between each pair of nodes in an undirected network. Gomory and Hu have studied this problem and have provided an efficient algorithm for its solution. We reexamine their procedure and generalize certain results of multi-terminal flow theory using well-known aspects of matroid theory. Additional implications afforded by this approach are also discussed.

1. Introduction

In their interesting paper [5] (see also [4, 6]) Gomory and Hu have considered the problem of determining the maximum flow value between each pair of nodes in a finite, undirected graph. This problem, known as the *multiterminal maximum flow problem*, has also been studied by Mayeda [9] and Chien [1]. In [3] Elmaghraby has examined the sensitivity of multi-terminal flows to changes in the capacity of a single edge in the graph. In the present paper we adopt the viewpoint of matroid theory and reexamine some basic results of multi-terminal flow theory in this more general, abstract setting. We begin with a brief summary of multi-terminal flow theory. In this discussion reader familiarity with the fundamental aspects of network flow theory, as set forth in [4], is presumed.

Assume given a finite, undirected graph (network) G. We will further require that G has neither loops nor multiple edges and that G is connected, though these latter assumptions are only for convenience of exposition. As usual, associated with each edge e of G is a nonnegative, real-valued *capacity* $c(e)$. We also have, for each unordered pair of nodes $\{x, y\}$ of G, a maximum flow value $v(\{x, y\})$[1] between x and y with respect to the given edge capacities. The real-valued, nonnegative function v is called the *flow function* for G. Notice that when G has n nodes v may be viewed as a function defined on the edges of K_n, the complete graph on n nodes.

Our primary concern is with the flow function v. One question of interest is that

* This research was partially supported by grant GK–42095 from the National Science Foundation to Yale University.

[1] The cumbersome notation is chosen to emphasize the fact that v is a function from the pairs of nodes of G to the nonnegative reals. The reason for this emphasis will become apparent in Section 2.

of *realizability*: When is a function the flow function of some graph? Gomory and Hu [5] have answered this question with the following characterization.

Theorem 1. *A function v from the edges of K_n to the nonnegative reals is the flow function of an n-node undirected network if and only if*

$$v(\{x_1, x_p\}) \geq \min\,[v(\{x_1, x_2\}), v(\{x_2, x_3\}), \ldots, v(\{x_{p-1}, x_p\})], \tag{1}$$

for any node sequence $x_1, x_2, \ldots, x_p$. □

Two networks which have the same flow function are termed *flow-equivalent*. An important consequence of (1) which becomes evident in the construction used to prove the sufficiency of these conditions is that every undirected network is flow-equivalent to a tree. Thus the flow function for a graph with n nodes assumes at most $n-1$ different values.

A second question of interest is the following: How does one efficiently determine the flow function for a given graph? Of course, one may construct the flow function for an n-node network by solving each of the $\binom{n}{2}$ maximum flow problems which correspond to all pairs of nodes in the network. However, since the flow function assumes at most $n-1$ distinct values, one might hope to do better. Gomory and Hu [5] have accomplished this by providing an elegant algorithm which determines the flow function by solving only $n-1$ maximum flow problems.

In order to describe their procedure we use the max-flow min-cut theorem of Ford and Fulkerson [4] to change emphasis slightly and view $v(\{x, y\})$ as the capacity of a minimum cut separating x and y. If sets X, $\bar{X}$ partition the nodes of G, we denote the corresponding cut by

$$(X, \bar{X}) = \{e = \{x, \bar{x}\} : x \in X, \bar{x} \in \bar{X} \quad \text{and } e \text{ is an edge of } G\}.$$

When each of the sets $X \cap Y$, $X \cap \bar{Y}$, $\bar{X} \cap Y$, $\bar{X} \cap \bar{Y}$ is nonempty, the two cuts $(X, \bar{X})$ and $(Y, \bar{Y})$ *cross* each other; otherwise these cuts are *non-crossing*. A family of cuts is termed *non-crossing* if each pair in the family is non-crossing. The following result which appears in [7] characterizes families of non-crossing cuts.

Lemma 1. *In a graph on n nodes, the families of $n-1$ non-crossing cuts correspond precisely to the spanning trees of K_n.* □

Certain of the minimum capacity cuts in a network also obey a non-crossing property. This is demonstrated in the following lemma, which is a simple consequence of the results of [5].

Lemma 2. *Suppose cuts $(X_1, \bar{X}_1), \ldots, (X_{k-1}, \bar{X}_{k-1})$ are non-crossing and $(X_i, \bar{X}_i)$ is a minimum capacity cut separating x_i and $\bar{x}_i$ for $1 \leq i \leq k-1$. Also assume that no $(X_i, \bar{X}_i)$ separates x_k and $\bar{x}_k$. Then there exists a minimum capacity cut $(X_k, \bar{X}_k)$ separating x_k and $\bar{x}_k$ which crosses no $(X_i, \bar{X}_i)$, $1 \leq i \leq k-1$.* □

The Gomory–Hu procedure is essentially an $(n-1)$-fold application of Lemma 2. One begins arbitrarily by choosing a pair of nodes $\{x_1, \bar{x}_1\}$, and determining a minimum cut $(X_1, \bar{X}_1)$ which separates x_1 and $\bar{x}_1$. At the kth stage ($k > 1$) one has non-crossing cuts $(X_1, \bar{X}_1), \ldots, (X_{k-1}, \bar{X}_{k-1})$, the ith being minimal for x_i and $\bar{x}_i$. If $k < n$ (the proof of) Lemma 1 shows that we may choose x_k and $\bar{x}_k$ which are separated by no $(X_i, \bar{X}_i)$, $1 \le i \le k-1$. One then determines $(X_k, \bar{X}_k)$ as described[2] in Lemma 2. The procedure continues until $k = n$ and termination occurs with $n-1$ non-crossing cuts.

By Lemma 1 these cuts correspond to a spanning tree T of K_n. T need not be a subgraph of G. Gomory and Hu call T the *cut-tree* for the graph G. This terminology reflects the fact that, for each pair $\{x, y\}$ of nodes of G, T determines *both* the flow value $v(\{x, y\})$ *and* a minimum cut of G which separates x and y. T specifies this information in the following manner: For any node pair $\{x, y\}$ corresponding to an edge of T, removal of that edge from T produces two subtrees with node sets, say, X_i and $\bar{X}_i$. Then the cut $(X_i, \bar{X}_i)$ is a cut of minimum capacity separating x and y in G; that is, $v(\{x, y\})$ is given by the capacity of $(X_i, \bar{X}_i)$. The cut $(X_i, \bar{X}_i)$ is, as the notation commemorates, one of those discovered by the algorithmic procedure of the preceding paragraph. Once we know $v(\{x, y\})$ for each $\{x, y\}$ corresponding to an edge of T, the remaining values for v may be determined by the relation

$$v(\{x, y\}) = \min\,[v(\{x_0, x_1\}), v(\{x_1, x_2\}), \ldots, v(\{x_{p-1}, x_p\})],$$

where $(x = x_0, x_1, x_2, \ldots, x_{p-1}, x_p = y)$ is the unique path from x to y in T.

2. Matroids

In the present section we summarize pertinent fundamental aspects of matroid theory. For a more thorough treatment the unfamiliar reader is referred to the works of Whitney [13], Tutte [12] and Minty [10].

A *matroid* $M = (E, \mathscr{C})$ is a finite set of *elements* $E = \{1, \ldots, n\}$ and a family $\mathscr{C}$ of nonempty subsets of E. Members of $\mathscr{C}$ are called *circuits* and they must satisfy the following two axioms:

(i) no circuit contains another,

(ii) if $C_1, C_2 \in \mathscr{C}$ with $e \in C_1 \cap C_2$ and $f \in C_1 \setminus C_2$, then there is some $C_3 \in \mathscr{C}$ for which $f \in C_3 \subseteq C_1 \cup C_2 \setminus \{e\}$.

A subset of E which contains a circuit is called *dependent*. A subset of E which contains no circuit is termed *independent* and a (set-wise) maximal independent set is called a *base*. It is clear that the minimal dependent subsets of E are precisely the

[2] Note that we have not specified *how* $(X_k, \bar{X}_k)$ is to be determined. This is accomplished by solving a maximum flow problem for x_k and $\bar{x}_k$ on a network obtained by suitably restricting G to ensure that $(X_k, \bar{X}_k)$ does not cross $(X_i, \bar{X}_i)$, $1 \le i \le k-1$.

circuits. A well-known consequence of axioms (i) and (ii) is that for any set $S \subseteq E$, every maximal independent subset of S is the same size. Another straightforward consequence is that for any base B and any $e \notin B$, $B \cup \{e\}$ contains a unique circuit C_e called a *fundamental* circuit relative to B; furthermore, $e \in C_e$ and for any $f \in C_e \setminus \{e\}$, $B' = B \cup \{e\} \setminus \{f\}$ is also a base.

Given a matroid $M = (E, \mathscr{C})$, let $\mathscr{C}^*$ denote the family of minimal, nonempty subsets $C^* \subseteq E$ which satisfy $|C^* \cap C| \neq 1$, for each $C \in \mathscr{C}$ ($|\cdot|$ is the cardinality function). It is not difficult to show that $M^* = (E, \mathscr{C}^*)$ is a matroid. Matroid M^* is called the *dual* of M; the circuits of M^* (members of $\mathscr{C}^*$) are called *cocircuits*. One can also show that the bases of M^* (called *cobases*) are simply the complements relative to E of the bases of M.

One standard example of a matroid comes from graph theory: Let E be the edge set of a finite, undirected graph G and let $\mathscr{C}$ denote the edge sets of the simple cycles in G. It is evident that $M = (E, \mathscr{C})$ satisfies axioms (i) and (ii). Such a matroid is called *graphic* and its dual is *cographic*. The cocircuits which define the dual matroid $M^* = (E, \mathscr{C}^*)$ are given by minimal cutsets for G; i.e., by minimal sets of edges whose removal increases the number of components of G. The bases of M are the edge sets of spanning forests in G. In Section 3 it is shown that the matroid M^* plays a central role in multi-terminal flow theory.

We now associate with each element $e \in E$ a weight $c(e)$ and consider the problem of determining a base of M which has maximum total weight. For graphic matroids this problem was treated by Kruskal [6] and Prim [11]. Kruskal's "greedy" algorithm for constructing a spanning forest of maximum weight is discussed for general matroids by Edmonds in [2], where it is shown to be a characterizing property of matroids. The following theorem provides necessary and sufficient conditions for a base to be of maximum weight. The theorem may be deduced from results in [2]; we provide an alternative proof based on matroid duality.

Theorem 2. *Let $M = (E, \mathscr{C})$ be a matroid with element weights $c(e)$, $e \in E$, and suppose B is a base of M. Then B is a maximum weight base if and only if*

$$e \notin B \implies c(e) \leq \min_{f \in C_e \setminus e} c(f), \tag{2}$$

where C_e is the fundamental circuit relative to B determined by e.

Proof. The necessity is clear, for if $c(e) > c(f)$ for some $f \in C_e \setminus \{e\}$, then $B' = B \cup \{e\} \setminus \{f\}$ is of larger weight than B. For the sufficiency suppose B_1 satisfies (2) and let B_2 be a base of maximum weight. We will show that B_1 and B_2 are of equal weight. If $B_1 = B_2$ we are done. Otherwise choose $e \in B_2 \setminus B_1$ and consider the fundamental circuit $C_e \subseteq B_1 \cup \{e\}$ and the fundamental cocircuit $C^*_e \subseteq (E \setminus B_2) \cup \{e\}$. Since $|C_e \cap C^*_e| \neq 1$, there is an element $f \neq e$ so that $f \in C_e \cap C^*_e$. Now $f \neq e$, so $f \in C_e$ implies $f \in B_1$ and $f \in C^*_e$ implies $f \notin B_2$. Thus $f \in B_1 \setminus B_2$. Consider the fundamental circuit $C_f \subseteq B_2 \cup \{f\}$. Since $|C_f \cap C^*_e| \neq 1$, there is an element $g \neq f$ so that $g \in C_1 \cap C_2^*$. Now $g \neq f$, so $g \in C_1$ implies $g \in B_2$. Thus $g \in B_2 \cap C_2^*$ and we conclude that $g = e$. Applying (2) first to B_1 and C_1 and then using the necessity to apply (2) to B_2 and C_2 shows that $c(e) = c(f)$. Thus $B_2' = B_2 \cup \{f\} \setminus \{e\}$ is also of maximum weight. Since $|B_2' \setminus B_1| < |B_2 \setminus B_1|$, iterative application of this argument shows that B_1 is of maximum weight. $\square$

3. Realizability conditions for matroids

Let x and y be two nodes of graph G and recall that $v(\{x, y\})$ represents the maximum flow between x and y in G. If edge $e = \{x, y\}$ is not present in G, we do not alter v for G by inserting e and defining $c(e) = 0$. Thus one may view $v(\{x, y\})$ as the minimum weight (capacity) of a cut containing e. Indeed, if G has n nodes and we define $c(e) = 0$ for edges of K_n not present in G, it is plain that the multi-terminal flow problem for G may be interpreted as follows: For each edge e of K_n determine a minimum weight cut containing e. In the present section we consider the same problem for arbitrary matroids and derive conditions analogous to (1) for the more general case.

Suppose $M = (E, \mathscr{C})$ is a matroid with nonnegative weights $c(e)$ for $e \in E$. Now let $v(e)$, $e \in E$, denote the minimum weight of a circuit which contains e; i.e.,

$$v(e) = \min_{C: e \in C \in \mathscr{C}} \sum (c(f) : f \in C)$$

(If element $e \in E$ is in no member of $\mathscr{C}$, define $v(e) = +\infty$.) We will call v the *minimum weight circuit function* for M. The following theorem provides conditions which must be satisfied by v.

Theorem 3. *Let $M = (E, \mathscr{C})$ be a matroid with nonnegative element weights $c(e)$, $e \in E$, and minimum weight circuit function v. Then for each $e \in E$,*

$$v(e) \geq \min_{f \in C^* \setminus \{e\}} v(f), \quad \forall C^* \in \mathscr{C}^* \quad \text{such that} \quad e \in C^*. \tag{3}$$

Proof. If e is in no circuit, then $v(e) = +\infty$ and (3) holds trivially; if e is a loop (single element circuit), then e is in no cocircuit and so (3) holds vacuously. Suppose C is a circuit of minimum weight containing e and let $e \in C^* \in \mathscr{C}^*$. Then $|C \cap C^*| \neq 1$ implies there is an $f \in C \cap C^*$ distinct from e. Since $f \in C$, $v(f) \leq \Sigma(c(g) : g \in C) = v(e)$. Thus v satisfies (3). $\square$

The conditions (3) are also sufficient in the following sense. For a given matroid $M = (E, \mathscr{C})$ and nonnegative, real-valued function v on E, call v *realizable* if there exists a nonnegative weight function c on E so that v is the minimum weight circuit function for M with respect to c. Theorem 4 below shows that the conditions (3) imply realizability. Together Theorems 3 and 4 constitute an appropriate generalization of Theorem 1 to arbitrary matroids. The proof of Theorem 4 follows closely the proof of sufficiency for Theorem 1 (see [4, 5, 6]).

Theorem 4. *Let $M = (E, \mathscr{C})$ be a matroid and let v be a nonnegative, real-valued function on E which satisfies (3). Then v is realizable as a minimum weight circuit function for M.*

Proof. Let B^* be a base of maximum weight for $M^* = (E, \mathscr{C}^*)$ with respect to the element weights $v(e)$, $e \in E$. Now define $c(e) = v(e)$ for $e \in B^*$ and $c(e) = 0$ for $e \in B = E \setminus B^*$. If $e \in B^*$, then $B \cup \{e\}$ contains a unique circuit C_e whose total weight with respect to c is $c(e)$. Since $c(f) \geq 0$ for all $f \in E$, C_e must be a minimum weight circuit for e. Note that $\sum(c(f) : f \in C_e) = c(e) = v(e)$. On the other hand if $e \in B$, then $B^* \cup \{e\}$ contains a unique cocircuit C_e^* and $e \in C_e^*$. From (2) (applied to B^*) and (3) it follows that $v(e) = \min_{f \in C_e^* \setminus e} v(f)$. Thus we may choose $f \in C_e^*$, $f \neq e$ for which $v(e) = v(f) = c(f)$. Now $B \cup \{f\}$ contains a unique circuit C_f whose total weight is $c(f)$. Since $|C_f \cap C_e^*| \neq 1$, we must have $e \in C_f$ also. Thus the minimum weight of a circuit containing e is no larger than $c(f)$. Suppose $e \in \bar{C} \in \mathscr{C}$. Since $|\bar{C} \cap C_e^*| \neq 1$, there is an element g so that $e \neq g \in \bar{C} \cap C_e^*$. Then it is straightforward to verify that

$$\sum(c(h) : h \in \bar{C}) \geq c(g) = v(g) \geq v(f) = c(f) = \sum(c(h) : h \in C_f).$$

Thus C_f is a minimum weight circuit for e. $\square$

For a given matroid we will say nonnegative weight functions c and c' are *equivalent* if they give rise to the same minimum weight circuit function v. The proof of Theorem 4 shows that to any given nonnegative weighting c for M there corresponds an equivalent weighting $c'(e)$, $e \in E$, for which the elements given nonzero weight are contained in a dual base B^*. When specialized to multi-terminal flows, this is simply the observation made earlier that each undirected network is flow-equivalent to a tree. Furthermore, to indentify minimum weight circuits *with respect to this equivalent weighting* c', one uses precisely the same rule as with the cut-tree: For $e \in B^*$, a minimum weight circuit for e is given by the circuit $C_e \subseteq B \cup \{e\}$ ($B = E \setminus B^*$); for $e \in B$, a minimum weight circuit is determined from among the $|C_e^*| - 1$ circuits $C_f \subseteq B \cup \{f\}$, for each $f \in C_e^* \setminus \{e\}$, where C_e^* is the cocircuit contained in $B^* \cup \{e\}$. As with multi-terminal flows, the following consequence is evident:

Corollary. *The number of distinct values assumed by a minimum weight circuit function for M is no greater than the size of a base for M^*.* $\square$

4. Discussion

Given that the realizability conditions for multi-terminal flows generalize directly for arbitrary matroids, it is natural to ask to what extent the algorithmic procedure of Gomory and Hu generalizes. Of particular interest would be the existence of a structure for matroids which determines not only the minimum weight circuit values but also the circuits themselves in a manner analogous to that of the cut-tree specified at the end of Section 1. Such an object does not exist in general[1], as is demonstrated by

Example: Consider the graphic matroid with edge weights as indicated in Fig. 1.

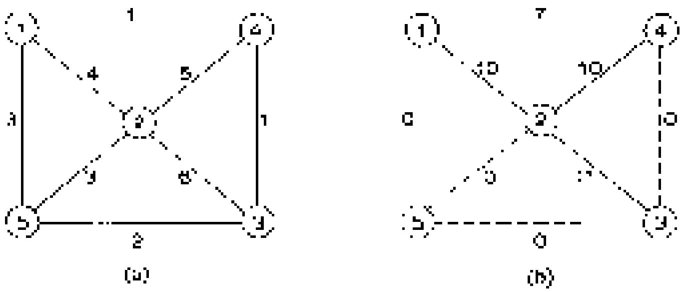


Fig. 1. (a) Original graph, (b) Equivalent weighting.

For this graph the minimum weight cycles for the edges are given by: cycle 14351, for edges {1,4}, {3,4}, {3,5} and {1,5}; 1241, for {2,4} and {1,2}; 1251, for {1,2} and {2,5}; 2352, for {2,3}. An equivalent weighting with nonzero weights only within a dual base is also given in Fig. 1(b). Note, however, that the minimum weight cycles determined by the equivalent weighting are not necessarily the same as those for the original graph; e.g., in the original graph cycle 1241 is of minimum weight for edge {2,4}, whereas with the equivalent weights, 24352 is the minimum weight cycle for {2,4}. It is not difficult to check that no equivalent weighting which is nonzero only on a dual base will determine the minimum weight cycles for the original graph. □

Thus in general one cannot expect to determine an equivalent weighting for a dual base which will play the role of the cut-tree in determining minimum weight circuits. This is not surprising — even for cographic matroids the cut tree need not be a subgraph of the original graph. A correct interpretation of this fact is that for a given matroid and element weights, there may exist no base whose fundamental circuits are minimum weight for the respective (out-of-base) elements which determine them. The latter is a consequence of the following proposition.

Proposition. *Let $M = (E, \mathscr{C})$ be a matroid with element weights $c(e)$, $e \in E$. Also let B be a base of M for which $e \notin B$ implies the fundamental circuit $C_e \subseteq B \cup \{e\}$ is a minimum weight circuit for e. If $e \in B$ is in a circuit, a minimum weight circuit for e is defined by*

[1] The author is indebted to R.E. Bixby for several helpful discussions concerning this point.

$$\min_{f \in C_e^*} \sum (c(g) : g \in C_f). \tag{4}$$

where C_e^* is the fundamental cocircuit relative to $B^* = E \setminus B$ determined by e, and C_f is the fundamental circuit relative to B determined by f.

Proof. First note that $e \in C_f$ for each C_f indicated by (4), else $|C_e^* \cap C_f| = 1$. Pick $e \in B$ and suppose $e \in C \in \mathscr{C}$. Since $|C_e^* \cap C| \neq 1$, there is an $f \in C \cap C_e^*$, $f \neq e$. Now $f \neq e$ and $f \in C_e^*$ imply that $f \notin B$. Consequently C_f is a minimum weight circuit for f. Thus $\sum (c(g) : g \in C_f) \leq \sum (c(g) : g \in C)$, which verifies (4). □

In the case of multi-terminal flows we recall that each edge e of the cut-tree T defines (by its removal from T) a minimum capacity cut C_e for e. Thus the above proposition provides a validation of the method described in Section 1 for determining $v(\{x, y\})$ when $\{x, y\}$ was not an edge of T. A further consequence of this proposition is the following simple proof of the existence of a cut-tree. We assume that each cut has a distinct capacity; if not, this may be achieved by perturbing edge capacities slightly, as described in [4]. Thus there will be *exactly* $n - 1$ distinct values for the minimum capacity cuts. The Gomory–Hu procedure described in Section 1 finds $n - 1$ non-crossing cuts, each of minimum capacity for some node pair. By Lemma 1 these cuts correspond to a tree T. Each edge of T is in only one of these $n - 1$ cuts, namely, the cut defined by the removal of that edge itself from T. Thus the edges of T satisfy the hypotheses of the above proposition, which implies that T is the cut-tree for G.

Finally, we remark that when G is a planar graph multi-terminal flow theory has implications for the shortest path problem. Associated with G is the dual graph G^* for which the cycles of G containing edge e of G correspond precisely to the cuts of G^* containing edge e of G^*. Thus for any edge e of G, the cut-tree T^* of G^* determines a minimum weight cycle which contains e. Such a cycle determines a shortest path from x to y in G, where $e = \{x, y\}$, by comparing the two possible paths from x to y around this cycle. Thus T^* provides a compact representation of all shortest path information for the edges of G.

References

[1] R.T. Chien, Synthesis of a communication net, IBM J. Res. Develop. 4 (1960) 311–320.

[2] J. Edmonds, Matroids and the greedy algorithm, *Math. Programming* 1 (1971) 127–136.

[3] S.E. Elmaghraby, Sensitivity analysis of multiterminal flow networks, *Oper. Res.* 12 (1964) 680–688.

[4] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, NJ, 1962).

[5] R.E. Gomory and T.C. Hu, Multi-terminal network flows, *J. Soc. Ind. and Appl. Math.* 9 (1961) 551–570.

[6] T.C. Hu, *Integer Programming and Network Flows* (Addison-Wesley, Reading, MA, 1969).

[7] T.C. Hu, Optimum communication spanning trees, MRC Technical Summary Report #1294, The University of Wisconsin-Madison (1973).

[8] J.B. Kruskal, Jr., On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.* 7 (1956) 48–50.
[9] W. Mayeda, Terminal and branch capacity matrices of a communication net, *IRE Trans. Circuit Theory*, Vol. CT-7 (1960) 251–269.
[10] G.J. Minty, On the axiomatic foundations of the theories of directed linear graphs, electrical networks, and network programming, *J. Math. Mech.* 15 (1966) 485–520.
[11] R.C. Prim, Shortest connection networks and some generalizations, *Bell Systems Tech. J.* 36 (1957) 1389–1401.
[12] W.T. Tutte, Lectures on matroids, *J. Res. Nat. Bur. Std.* B 69 (1965) 1–47.
[13] H. Whitney, On the abstract properties of linear dependence, *Am. J. Math.* 57 (1935) 509–533.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 1 (1977) 527–538

VALID INEQUALITIES, COVERING PROBLEMS AND DISCRETE DYNAMIC PROGRAMS

Laurence A. WOLSEY

Center for Operations Research & Econometrics, Université Catholique de Louvain, Belgium

Various discrete optimization problems such as the integer and 0–1 programming problems, and the travelling salesman problem have been represented as discrete dynamic programming, or network problems. We show how such representations lead naturally to a characterization of the valid inequalities for the feasible solution sets Q of such problems. In particular we obtain polytopes Π of valid inequalities having the facets of Q among their extreme points. In addition the problems of "packing" or "covering" with feasible solutions to the discrete problem have natural network representations, which are the duals of problems over Π.

Reversing the approach, any special properties of the valid inequalities can in turn be used to give new formulations of the corresponding network problems. In particular this allows a reformulation of the "minimum equivalent knapsack inequality" problem, and the "cutting stock" problem.

1. Introduction

The characterization of valid inequalities for various combinatorial problems has been the subject of much recent research. The motivation is in part practical, stronger valid inequalities giving better bounds and cuts, and partly theoretical in the belief that a better understanding of the underlying structures will eventually lead to improved algorithms.

In this paper we examine the question of characterizing such inequalities for the class of combinatorial problems that can be viewed as discrete dynamic programs, and the consequences of the fact that this question can be reinterpreted in terms of a "covering" problem.

Various combinatorial problems (P_0)

$$\max\{cx \mid x \in Q \subset \mathbf{R}^n\}$$

can be viewed as discrete dynamic programs, or longest route network problems. Here we show that such a viewpoint provides useful information on two related problems (P_1)

Find a polytope Π such that $(\pi; \pi_0) \in \Pi$ if and only if $\pi x \leq \pi_0$ is a non-trivial valid inequality for Q,

and (P_2)

(The Covering Problem) $\min\{1 \cdot y \mid By \geq w, y \geq 0\}$ where the columns of B are vectors representing the feasible points of Q, and w is a nonnegative integer vector.

Relationships between these three problems have been demonstrated in various special cases, but do not seem to have been fully exploited.

For instance by duality (P_1, P_3)

$$\begin{aligned} \min\{1\cdot y \mid By \geq w, y \geq 0\} &= \max\{w\pi \mid \pi B \leq 1, \pi \geq 0\} \\ &= \max\{w\pi \mid (\pi, \pi_0) \in \Gamma, \pi_0 \leq 1, \pi \geq 0\} \end{aligned}$$

See [4] on antiblocking polyhedra where this duality is used but very special representations of Γ are sought.

(P_0, P_2) The problem (P_0): $\max\{\pi x \mid x \in Q\}$ can be formulated as a DP recursion or as a network flow problem. We claim that Γ is obtained by constraining (π, π_0) to be dual-feasible for the network flow problem. Alternatively the constraints of Γ are directly evident from the DP recursion. See [1] and [6] where polytopes Γ closely resembling the DP recursion have been obtained.

(P_0, P_3) The dual of $\max\{w\pi \mid (\pi, \pi_0) \in \Gamma, \pi_0 \leq 1, \pi \geq 0\}$ gives a representation of the covering problem on the network associated with (P_0). Unlike (P_2) this problem is not totally unimodular due to capacity constraints involving several arcs simultaneously.

Below we shall look at several examples so as to demonstrate the relationships. In Section 2 we look at the 0–1 monotone problem in some detail. Here the representation of Γ has two apparent advantages over other suggested representations, simplicity, and a limited number $O(n\beta)$ of constraints and variables where β is the number of DP states. For the special case of the 0–1 knapsack problem, we show that the "minimum equivalent inequality problem" is equivalent to a variety of covering problems.

In Section 3 we look at the integer monotone problem. Here although a good deal about representations of Γ is known [2], no one has apparently looked at the corresponding representation of (P_2). Both the natural network representation, and a representation based on a "minimal" Γ appear new, and appear to have advantages over the standard column generating formulations of the "cutting stock" problem.

Finally, in Section 4, we mention briefly two other problems amenable to treatment in this way.

To close this section we give two general definitions.

Definition 1.1. The inequality:

$$\sum_{j=1}^{n} \pi_j x_j \leq \pi_0 \tag{1}$$

is said to be a *valid inequality* denoted $(\pi; \pi_0)$ for Q if every feasible point in Q satisfies the inequality. A valid inequality is a *facet* of Q if $\exists n$ affinely independent points of Q satisfying it with equality.

Definition 1.2. A set of nonnegative integer vectors Y is *monotone* if whenever x' is a nonnegative integer vector such that $x' \leq x$, and $x \in Y$, then $x' \in Y$.

Throughout the paper we shall only be concerned with valid inequalities with $\pi_0 \neq 0$. (If Q is monotone, this only essentially eliminates the trivial inequalities $x_j \geq 0$).

2. Valid inequalities for 0–1 monotone polytopes

Here we consider the 0–1 monotone polytope, from the viewpoint that, given a linear objective, we obtain a problem amenable to solution by discrete dynamic programming, or as a longest path network problem.

We consider in particular the set Q:

$$\sum_{j=1}^{n} a_j x_j \leq b, \quad x_j \in \{0, 1\}$$

where $\{a_j\}_{j=1}^{n}, b \in \mathbf{Z}_+^m$ (the set of m-dimensional nonnegative integer column vectors).

Clearly the set Q above is monotone.

Now we shall define certain standard terms used in dynamic programming. Let

$$Q_r(\lambda) = \left\{ x \,\middle|\, \sum_{j=1}^{r} a_j x_j \leq \lambda, x_j \in \{0, 1\} \right\},$$

$$G_r(\lambda) = \max \left\{ \sum_{j=1}^{r} \pi_j x_j \,\middle|\, x \in Q_r(\lambda) \right\},$$

where the terms are only defined for $0 \leq r \leq n$, $\lambda \in \mathbf{Z}_+^m$ with $\lambda \leq b$.

Note that $Q = Q_n(b)$ and that $G_r(\lambda) = \max \{G_{r-1}(\lambda), G_{r-1}(\lambda - a_r) + \pi_r\}$, where $G_0(\lambda) = 0$ for $0 \leq \lambda \leq b$, and any expression containing an undefined term is ignored.

Also (1) is a valid inequality for Q if and only if $\pi_0 \geq G_n(b)$. It is *tight* if $\pi_0 = G_n(b)$.

Now let us write (P_0): $\max \{\pi x \mid x \in Q\}$ as a network flow problem, with nodes (r, λ) for $0 \leq r \leq n$, and $0 \leq \lambda \leq b$, edges $[(r-1, \lambda - a_r), (r, \lambda)]$ and $[(r-1, \lambda), (r, \lambda)]$ containing flows $\xi_r(\lambda)$, $\eta_r(\lambda)$ respectively $r = 1, 2, \ldots, n$, $0 \leq \lambda \leq b$ if both endpoints of the edge are legitimate nodes.

Proposition 2.1. *Problem* (P_0) *is equivalent to the totally unimodular flow problem* (FP_0):

$$G_n(b) = \max \sum_r \sum_\lambda \pi_r \xi_r(\lambda)$$

$$\text{s.t.} \quad \xi_r(\lambda) + \eta_r(\lambda) - \xi_{r+1}(\lambda + a_{r+1}) - \eta_{r+1}(\lambda) = 0$$

$$r = 1, 2, \ldots, n-1 \quad 0 \leq \lambda \leq b$$

$$\xi_1(\lambda) - \eta_1(\lambda) = 0 \quad 0 \leq \lambda \leq b$$

$$\xi_n(b) + \eta_n(b) = 1$$

$$\xi_r(\lambda), \eta_r(\lambda) \geq 0 \quad r = 1, 2, \ldots, n; \quad 0 \leq \lambda \leq b.$$

Proof. We note that with this choice of notation, an $x \in Q$ with $ax = b - \mu$ corresponds to a path in the network from $(0, \mu)$ to (n, b) or a feasible solution of (FP_0). Hence $G_n(b) \le \sum_\lambda \sum_r \pi_r \xi_r(\lambda)$. Conversely the linear program has an optimal solution which is integer. It therefore corresponds to a path from $(0, \mu)$ to (n, b), and an $x \in Q$ with $ax = b - \mu$. Hence $\sum_\lambda \sum_r \pi_r \xi_r(\lambda) \le G_n(b)$.

Theorem 2.2. *$(\pi; \pi_0)$ is a valid inequality for Q if and only if there exist values $\theta_r(\lambda)$, $r = 1, 2, \ldots, n$, $0 \le \lambda \le b$ such that $(\pi; \theta_r(\lambda)) \in \Gamma$ with $\pi_0 = \theta_n(b)$ where:*

$$\Gamma = \left\{ (\pi; \theta_r(\lambda)) \;\middle|\; \begin{array}{ll} \theta_r(\lambda) - \theta_{r-1}(\lambda - a_r) - \pi_r & \ge 0 \\ \\ \theta_r(\lambda) - \theta_{r-1}(\lambda) & \ge 0 \end{array} \right.$$

$r = 1, 2, \ldots, n$, $0 \le \lambda \le b$, where again undefined terms vanish i.e. when $r = 1$, the constraints become $\theta_1(\lambda) - \pi_1 \ge 0$; $\theta_1(\lambda) \ge 0$ when $b \ge \lambda \ge a_1$, and reduce to $\theta_r(\lambda) \ge \theta_{r-1}(\lambda)$ when $a_r < \lambda$.

Proof. $(\pi; \theta_n(b))$ is a valid inequality for Q if and only if $\theta_n(b) \ge G_n(b)$. Taking the dual of (FP_0) we obtain $\{\min \theta_n(b) \mid (\pi; \theta_r(\lambda)) \in \Gamma\} = G_n(b)$, and hence if $(\pi, \theta_r(\lambda)) \in \Gamma$ with $\pi_0 = \theta_n(b)$, then $(\pi; \pi_0)$ is valid for Q. The converse is immediate taking $\theta_r(\lambda) = G_r(\lambda)$.

Remark 2.3. Replacing $G_r(\lambda)$ by $\theta_r(\lambda)$ in the DP recursion, we see that Γ can be obtained directly.

Theorem 2.4. *If $\sum_{j=1}^n \pi_j x_j \le \pi_0$ is a non-trivial facet of Q, then $(\pi, G_r(\lambda))$ is an extreme point of Γ with $\pi_0 = G_n(b)$.*

Proof. Suppose not. Then

$$(\pi, G_r(\lambda)) = \tfrac{1}{2}(\pi^1, \theta_r^1(\lambda)) + \tfrac{1}{2}(\pi^2, \theta_r^2(\lambda)).$$

Case (a). $\pi^i \ne \pi$, $i = 1, 2$. This contradicts the fact that a non-trivial facet of Q is extreme among the valid inequalities for Q.

Case (b). $\pi^1 = \pi^2 = \pi$. Then as $(\pi, \theta_r^i(\lambda)) \in \Gamma$, $\theta_r^i(\lambda) \ge G_r(\lambda)$, $i = 1, 2$. However $\frac{1}{2}\theta_r^1(\lambda) + \frac{1}{2}\theta_r^2(\lambda) = G_r(\lambda)$, and hence $\theta_r^i(\lambda) = G_r(\lambda)$, $i = 1, 2$ $\forall r, \lambda$, contradicting the hypothesis that $(\pi^1, \theta_r^1(\lambda))$ and $(\pi^2, \theta_r^2(\lambda))$ are distinct.

Therefore we have shown that the extreme points of Γ, restricted to the variables π, $\theta_n(b)$ include the non-trivial facets of Q.

Remark 2.5. We note also that the polytope Γ is very easy to describe and has at most $(n - 1)\beta$ variables and $2n\beta$ constraints where $\beta = \prod_{i=1}^m (b_i + 1)$.

An alternative characterization of valid inequalities for knapsack problems is given in [2, 7], based in part upon the total ordering among the variables. Although it is efficient for small values of b, the number of constraints in the resulting polytope grows exponentially with b.

Consider now the covering problem ($\mathbf{P}_2$), and in particular its representation in the form $\max\{w\pi \mid (\pi, \theta_r(\lambda)) \in \Gamma, \theta_n(b) \le 1, \pi \ge 0\}$. Taking its dual we obtain the network flow problem

$$
\begin{aligned}
\min\ & Z_0\\
& \xi_r(\lambda) - \eta_r(\lambda) - \xi_{r-1}(\lambda + a_{r-1}) - \eta_{r-1}(\lambda) = 0,\\
& \xi_0(\lambda) + \eta_0(\lambda) = 0,\\
& \xi_n(b) + \eta_n(b) - Z_0 = 0,\\
\sum_{\lambda} & \xi_r(\lambda) \ge w_r,\\
& \xi_r(\lambda), \eta_r(\lambda) \ge 0, Z_0 \ge 0.
\end{aligned}
$$

This problem involves the same network as in (FP_0) where the first three constraint sets represent flow feasibility constraints, but the last "covering" set of constraints imposes a minimum aggregate flow over certain subsets of the arcs, and destroys the property of total unimodularity.

An application to find "minimum equivalent knapsack inequalities"

Given a single linear inequality (**L**)

$$\sum_{j=1}^{n} a_j x_j \le a_0, \quad x_j \in \{0, 1\}$$

where $a_j > 0$, we consider the problem of finding a linear inequality $\sum_{j=1}^n b_j x_j \le b_0$ which is

(i) valid for (L),

(ii) $\sum_{j=1}^n b_j y_j \ge b_0 + 1$ for all 0–1 points y not lying in (**L**),

(iii) for which b_0 is minimum.

We restrict ourselves without loss of generality, (see [2]), to inequalities (L) for which x is feasible if and only if $e - x$ is infeasible, where $e = (1, 1, 1, \ldots, 1)^T$. Now it is known that with this restriction the extreme points of the polyhedron of valid inequalities defined by (i), (ii) satisfy $\sum_{j=1}^n b_j = 2b_0 + 1$, and that this equality plus (i) implies (ii), see [2, 10, 11].

Therefore the problem can be reduced to (R_0)

$$\min\left\{\pi_0 \mid \pi B \le \pi_0 e, \sum_{j=1}^{n} \pi_j = 2\pi_0 + 1, \pi_j \ge 0\right\}$$

where B as in the introduction is the matrix having the 0–1 feasible solutions to (**L**) as columns.

Below we shall use our results to derive several reformulations of ($\mathbf{R}_0$). Using the characterization of Γ we obtain ($\mathbf{R}_1$):

$$\min\left\{\theta_n(a_0) : (\pi, \theta(\lambda)) \in \Gamma, \sum_{j=1}^{n} \pi_j = 2\theta_n(a_0) + 1, \pi_j \ge 0\right\}$$

or its network dual (R_2)

$$\max\ Z_0,$$

$$\begin{aligned}
&\xi_i(\lambda)+\eta_i(\lambda)-\xi_{i+1}(\lambda+a_{i+1})-\eta_{i+1}(\lambda)=0,\\
&\xi_n(\lambda)-\eta_n(\lambda)=0,\\
&\xi_n(a_0)-\eta_n(a_0)-2Z_0=1,\\
-&\sum_\lambda \xi_1(\lambda)-Z_0\geq 0,\\
&\xi_i(\lambda),\ \eta_i(\lambda)\geq 0.
\end{aligned}$$

In terms of solutions of (T_i), (R_2) is evidently equivalent to (R_3):

$$\begin{aligned}
\max\ & Z_0,\\
& e\cdot y=1+2Z_0,\\
& By\geq Z_0e,\\
& y\geq 0,
\end{aligned}$$

where the variables y can be thought of as the weights given to the different paths from $(0,0)$ to (n,a_0). Hence (R_3) is a special covering problem (also obtainable directly as the dual of (R_0)). Finally substituting for Z_0 in (R_3) gives (R_4):

$$\begin{aligned}
\max\ & \tfrac{1}{2}e\cdot y-\tfrac{1}{2},\\
\text{s.t.}\ & (E-2B)y\leq e,\\
& y\geq 0,
\end{aligned}$$

a special packing problem whose matrix $E-2B$ has entries ± 1, where E is a matrix of all 1's.

Alternatively changing the normalization, and replacing (ii) by $\sum_{j=1}^n b_jy_j\geq b_0$, $b_0=2$ and (iii) by $\max\sum_{j=1}^n b_j$ we can replace (R_0) by (R_5)

$$\zeta=\max\{\pi\cdot e \mid \pi B\leq e,\ \pi\geq 0\}.$$

This follows from two observations. First that $\zeta\geq 2$, and any valid inequality with $\pi\cdot e\geq 2$ is a representation of the inequality. Second that if π^* is an optimal solution of (R_5) with $\pi^*e=2+1/t$, $t>0$, then $t\pi^*$ is an optimal solution of (R_0), and conversely.

Taking its dual we obtain a last reformulation (R_6):

$$\begin{aligned}
\zeta=\min\ & e\cdot y\\
& By\geq e\\
& y\geq 0
\end{aligned}$$

the most basic covering problem.

Remark 2.5 suggests that formulations (R_3) and (R_4) may be advantageous computationally. See [2] for computational results using a formulation derived from (R_6).

Example.

$$6x_1 + 5x_2 + 4x_3 + 4x_4 + 2x_5 \le 10, \quad x_j \in \{0, 1\}$$

$$\pi_n(a_n) = Z_n = 3, \qquad \sum_{j=1}^{5} \pi_j = c \cdot y = 7, \qquad \xi = \tfrac{3}{7}.$$

Minimum Equivalent Inequality

$$2x_1 + 2x_2 + x_3 + x_4 + x_5 \le 3, \quad x_j \in \{0, 1\}.$$

Cover {1, 3}{2, 3}{1, 4}{2, 4}{1, 5}{2, 5}{3, 4, 5}.
These 7 solutions together make use of each variable at least 3 times.

3. Valid inequalities for integer monotone polytopes

Here we consider the set Q:

$$\sum_{j=1}^{n} a_j x_j \le b$$
$$x_j \ge 0 \text{ and integer}$$

where $a_j, b \in Z^1_+$.

For this problem all the representations of the valid inequalities that we present are known. We stress however that they also derive from the DP or network structure. The main emphasis below is on (P_3) the covering, or "cutting stock" problem, and we use the characterizations of Γ and of a subset of the valid inequalities including the maximal and extreme valid inequalities to obtain two different formulations of (P_3).

Defining $G(\lambda) = \max\{\sum_{j=1}^{n} \pi_j x_j \mid \sum_{j=1}^{n} a_j x_j \le \lambda, x_j \ge 0 \text{ and integer}\}$, we obtain from dynamic programming the recursion

$$G(\lambda) = \max\left[0, \max_{j=1,\dots,n} \{G(\lambda - a_j) + \pi_j\}\right],$$

where $G(\lambda)$ is defined for $\lambda \le b$, $\lambda \in Z^1_+$, and undefined terms are ignored. This leads immediately to the network problem:

$$G(b) = \max \sum_{\lambda} \sum_{j} \pi_j \xi_{\lambda - a_j, \lambda}$$

$$\begin{aligned}
\text{s.t.} \quad & & -\sum_j \xi_{0,a_j} &\ge 0 \\
& \sum_j \xi_{\lambda - a_j, \lambda} & -\sum_j \xi_{\lambda, \lambda + a_j} &\ge 0 \quad 0 < \lambda < b, \ \lambda \in Z_+ \\
& \sum_j \xi_{b - a_j, b} & &\le 1 \\
& & \xi_{\lambda, \lambda + a_j} &\ge 0
\end{aligned}$$

Lemma 3.1. *$(\pi;\pi_0)$ is valid for Q if and only if there exist values $\theta(\lambda)$, $\lambda \le b$, $\lambda \in Z^m_+$ such that $(\pi, \theta(\lambda)) \in \Gamma$ with $\pi_0 = \theta(b)$, where*

$$\Gamma = \{(\pi, \theta(\lambda)) \mid \theta(\lambda) - \theta(\lambda - a_j) \ge \pi_j, \ \theta(\lambda) \ge 0\}$$

Proof. From the dual of the above problem we have that $G(b) = \min\{\theta(b) \mid (\pi, \theta(\lambda)) \in \Gamma\}$. Conversely $(\pi, G(\lambda)) \in \Gamma$.

We see also that once again Γ could have been written down directly from the DP recursion.

Looking now at the covering problem (P$_2$) we have (R$_2$):

$$\max\{w \cdot \pi \mid (\pi, \theta(\lambda)) \in \Gamma, \ \theta(b) \le 1, \ \pi \ge 0\}.$$

We note that this problem has $n + \beta$ variables and $n\beta$ constraints, and hence has far fewer constraints than in the standard column generation formulation.

Its dual is (R$_3$):

$$
\begin{aligned}
\min \quad & Z_0 \\
\text{s.t.} \quad & \qquad\qquad \sum_j \xi_{0,a_j} \ge 0, \\
& \sum_j \xi_{\lambda,a_j} - \sum_j \xi_{\lambda+a_j,a_j} \ge 0, \\
& -\sum_j \xi_{b,a_j} \qquad - Z_0 \ge 0, \\
& \sum_\lambda \xi_{\lambda,a_j} \qquad\qquad \ge w_j, \\
& \qquad \xi_{\lambda,a_j} \ge 0, \ Z_0 \ge 0.
\end{aligned}
$$

This is again a network flow problem with additional constraints, where ξ_{λ,a_j} is the number of cutting patterns (solutions of Q) with a piece of length a_j cut between $\lambda - a_j$ and λ ($x_j - 1$ when in state $\lambda - a_j$).

Now we consider the possibility of replacing Γ in problem (R$_1$) by some other polytope of valid inequalities.

Theorem 3.2. [1] *Every maximal inequality*

$$(\pi;\pi_0) = (\theta(a_j);\theta(b))$$

of Q lies in $\Gamma^* = \{\theta(\lambda) \mid \theta(\lambda) \ge \theta(\lambda - a_j) + \theta(a_j), \ \theta(\lambda) \ge 0\}$ and conversely if $\theta(\lambda) \in \Gamma^*$, then $(\theta(a_j);\theta(b))$ is valid for Q. In addition the extreme points of Γ^* include the non-trivial facets of Q.

The above polytope is suggested by noting that an inequality is maximal for Q

only if $\pi_i = G(a_i)$, and that $(\pi, G(\lambda)) \subset \Gamma$, so that necessarily the maximal inequalities are in Γ^*.

Now as the optimal solution to problem (P_3) and hence (R_1) corresponds to a non-trivial facet of Q, we obtain the equivalent problem (R_3):

$$\max \sum_{i=1}^{n} w_i\theta(a_i)$$

$$\theta(\lambda) \subset \Gamma^*, \ \theta(b) \leq 1.$$

Associate the dual variables $\eta_{\lambda,\mu}$ with the constraint $-\theta(\lambda+\mu)+\theta(\lambda)+\theta(\mu) \geq 0$ where either λ or μ or both equal a_i for some $i = 1, 2, \ldots, n$ and if "$\preccurlyeq$" is some ordering of the elements of $0 \leq \lambda \leq b$, $\lambda \in Z_+$, $\eta_{\lambda,\mu}$ is only defined for $\lambda \preccurlyeq \mu$.

This problem then has as dual (R_4):

$$\min Z_0$$

node a_i:

$$\sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} \geq w_i;$$

node $\mu \neq a_i$:

$$\sum_{[illegible]} \eta_{[illegible]} + \sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} \geq 0;$$

node b:

$$-\sum_{[illegible]} \eta_{[illegible]} - \sum_{[illegible]} \eta_{[illegible]} + Z_0 \geq 0;$$

$$\eta_{\lambda,\mu} \geq 0, \ Z_0 \geq 0.$$

Here we can interpret $\eta_{\lambda,\mu}$ as the number of paths passing through the nodes λ and $\lambda+\mu$, and using a single arc from λ to $\lambda+\mu$. Alternatively it is the number of cutting patterns having a single piece of size μ in position $(\lambda, \lambda+\mu)$.

Looking at the inequalities in (R_4), the first term counts the number of single pieces of size a_i that occur in the cutting patterns in any but the first i.e. $(0, a_i)$ position. The remaining three terms count the number of single pieces that occur in the first position, in particular the number of pieces with a cut in position a_i (term 2) less the number of pieces containing a cut between 0 and a_i (term 3 + term 4). The final constraint counts the total number of patterns used Z_0.

(R_4) can also be obtained by eliminating the variables $\xi_{\alpha,\lambda}$ in (R_2).

Example.

$$Q = \{x \mid 2x_1 + 3x_2 + 5x_3 = 6, \ x_i \geq 0 \text{ and integer}\}.$$

$$(w_1, w_2, w_3) = (8, 9, 4).$$

(R_4)

$$
\begin{array}{lllllllll}
\max & & 8\theta(2) & -3\theta(3) & & +4\theta(5) & & & \\
 & \theta(1) & -\theta(2) & -\theta(3) & & & & \leq 0, & \eta_{12} \\
 & & 2\theta(2) & & -\theta(4) & & & \leq 0, & \eta_{22} \\
 & & \theta(2) & +\theta(3) & & -\theta(5) & & \leq 0, & \eta_{23} \\
 & & \theta(2) & & +\theta(4) & & -\theta(6) & \leq 0, & \eta_{24} \\
 & \theta(1) & & -\theta(3) & -\theta(4) & & & \leq 0, & \eta_{34} \\
 & & & 2\theta(3) & & & \theta(6) & \leq 0, & \eta_{35} \\
 & \theta(1) & & & & +\theta(5) & -\theta(6) & \leq 0, & \eta_{45} \\
 & & & & & & \theta(6) & \leq 1, & Z_0 \\
 & & & & & & \theta(\lambda) & \geq 0, & Z_0 \geq 0.
\end{array}
$$

Remark. The formulations (R_1)–(R_4) can be used in column generation procedures, as in the standard Gilmore and Gomory (1961) approach. The new column at iteration k is generated by taking the current variables π^k and solving: $\max\{\pi^k x \mid x \in Q\}$. Supposing x^k is the optimal solution, either $\pi^k x^k = 1$ and the algorithm terminates, or $\pi^k x^k > 1$, and at least one of the constraints: $\theta(\lambda) + \theta(a_r) \geq \theta(\lambda + a_r)$ is violated for some r with $x_r^k = 1$, and can be added to generate the new problem at iteration $k+1$.

Example (cont). Starting from

$$
\begin{array}{llllll}
\max & 8\theta(2) - 3\theta(3) - 4\theta(5) & & & & \\
\text{s.t.} & \theta(1) + \theta(2) & -\theta(3) & & & \leq 0, \\
 & & 2\theta(3) & & -\theta(6) & \leq 0, \\
 & \theta(1) & & +\theta(5) & -\theta(6) & \leq 0, \\
 & & & & \theta(6) & \leq 1, \\
 & & & & \theta(\lambda) & \geq 0,
\end{array}
$$

we solve the LP to obtain

$$\pi^1 = \theta = (0, \tfrac{1}{2}, \tfrac{1}{2}, 0, \tfrac{1}{2}, 1).$$

Solving

$$\max \tfrac{1}{2}x_1 + \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3$$

$$\text{s.t. } 2x_1 - 3x_2 - 5x_3 \leq 6, \quad x_j \in \{0, 1\},$$

we obtain the optimal solution $x^1 = (3, 0, 0)$ with $\pi^1 x^1 = \tfrac{3}{2} > 1$.

The solution x^1 indicates immediately that at least one of the constraints

$$2\theta(2) - \theta(4) \geq 0$$

$$\theta(2) + \theta(4) - \theta(6) \geq 0$$

is violated.

Adding these constraints and resolving the LP, we obtain $\pi^2 = \theta = (0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 1, \ldots)$. Solving

$$\max \tfrac{1}{2}x_1 + \tfrac{1}{2}x_2 + 1x_3,$$

$$\text{s.t. } 2x_1 + 4x_2 + 6x_3 \leq 6, \qquad x_j \in \{0, 1\},$$

we obtain an optimal solution $x^3 = (0, 0, 1)$ with $\pi^2 x^3 = 1$ and therefore π^2 is the optimal solution.

4. Further dynamic polytopes

Various other dynamic programming recursions generate polytopes of valid inequalities in a natural way. We consider two examples.

(1) *The travelling salesman problem*

A well-known recursion for the problem [8] is the following:

$$f(S, j) = \min\{c_{ij} + f(S - j, i)\}$$

over all $i \in S - j$, where c_{ij} is the distance from i to j, and $f(S, j)$ is the minimum length path which starts at vertex 1, visits all vertices in S, and terminates at $j \in S$, where $S \subseteq N = \{1, 2, \ldots, n\}$. Q is the set of all tours and $G(S, j)$ the shortest length path with arc lengths π_{ij}.

We obtain the dynamic polytope:

$$\Gamma = \{(\pi_{ij}, \theta(S, j)) \mid \theta(S, j) \leq \pi_{ij} + \theta(S - j, i) \forall i \in S - j, \pi_{ij} \geq 0, \theta(S, j) \geq 0\},$$

for the valid inequalities $\sum\sum \pi_{ij} x_{ij} \geq \pi_0$ for Q (denoted (π_{ij}, π_0)).

Lemma 4.1. *(π_{ij}, π_0) is a valid inequality for Q if and only if there exist values $\theta(S, j)$ such that $(\pi_{ij}, \theta(S, j)) \in \Gamma$ with $\pi_0 = \theta(N, 1)$. If (π_{ij}, π_0) is extreme among the valid inequalities for Q, then $(\pi_{ij}, G(S, j))$ is extreme in Γ with $\pi_0 = G(N, 1)$.*

(2) *Equality constrained 0-1 problems*

Taking $Q = \{x \mid \sum_{j=1}^{n} a_j x_j = b, x_j \in \{0, 1\}\}$ with $\{a_j\}_{j=1}^{n}$, $b \in \mathbf{Z}_+$, and $Q_r(\lambda) = \{x \mid \sum_{j=1}^{r} a_j x_j = \lambda, x_j \in \{0, 1\}\}$, and $G_r(\lambda) = \max\{\sum_{j=1}^{r} \pi_j x_j \mid x \in Q_r(\lambda)\}$, we obtain the dynamic polytope

$$\Gamma = \left\{ (\pi, \theta_r(\lambda)) \;\middle|\; \begin{array}{ll} \theta_r(\lambda) \geq \theta_{r-1}(\lambda - a_r) + \pi_r & r = 1, \ldots, n \\ \theta_r(\lambda) \geq \theta_{r-1}(\lambda) & \lambda \leq b, \lambda \in \mathbf{Z}_+ \\ \theta_0(0) \geq 0 & \end{array} \right.$$

with $\theta_r(\lambda)$ defined for $r = 0, 1, \ldots, \lambda \le b$, $\lambda \in \mathbf{Z}_m$, generating valid inequalities: $\sum \pi_j x_j \le \theta_n(b)$ for Q where $\pi_j \to +\infty$ if there is no solution with $x_j = 1$, or $\theta_r(\lambda) \to -\infty$ if $Q_r(\lambda)$ has no solution.

References

[1] J.A. Araoz-Durand, Polyhedral Neopolarities, Ph.D. thesis, University of Waterloo, Faculty of Mathematics, Department of Applied Analysis and Computer Science (1973).

[2] G.H. Bradley, P.L. Hammer and L.A. Wolsey, Coefficient Reduction for Inequalities in 0–1 Variables, *Math. Programming*, 7 (1974) 263–282.

[3] N. Christofides and S. Eilon, The Loading Problem, *Management Sci.*, 17 (1971) 259–268.

[4] D.R. Fulkerson, Blocking and Antiblocking Pairs of Polyhedra, *Math. Programming*, 1 (1971) 168–194.

[5] P. Gilmore and R.E. Gomory, A Linear Programming Approach to the Cutting Stock Problem I, *Operations Res.*, 9 (1961) 849–859.

[6] R.E. Gomory, Some Polyhedra Related to Combinatorial Problems, *Linear Algebra Appl.*, 2 (1969) 451–558.

[7] P.L. Hammer and U.N. Peled, Computing Low Capacity 0–1 Knapsack Polytopes, CORR No. 75-4, University of Waterloo, Faculty of Mathematics (1975).

[8] M. Held and R.M. Karp, A Dynamic Programming Approach to Sequencing Problems, *SIAM Appl. Math.*, 10 (1962) 196–210.

[9] T.L. Morin and R.E. Marsten, Branch and Bound Strategies for Dynamic Programming, Discussion Paper No. 106, Northwestern University, Center for Mathematical Studies in Economics and Management Science (1974).

[10] S. Muroga, *Threshold Logic and its Applications*, (Wiley, New York, 1971).

[11] B. Peleg, On Weights of Constant-Sum Majority Games, *SIAM J. Appl. Math.*, 20 (1968) 527–532.

Annals of Discrete Mathematics 1 (1977) 579–590
© North-Holland Publishing Company

SOME PARTIAL ORDERS RELATED TO BOOLEAN OPTIMIZATION AND THE GREEDY ALGORITHM

Uwe ZIMMERMANN

Mathematisches Institut, Universität Köln, 5 Köln 41, Weyertal 86–90, F.R.G.

For $\mathbf{B} = \{0, 1\}$ and ordered sets $(H, \leq)$ the objective $f : \mathbf{B}^n \to H$ shall be maximized under the restriction $x \in S \subseteq \mathbf{B}^n$. The Greedy algorithm can be formulated for this problem without difficulties. The question is for which objectives f and which restrictions S one can use the algorithm to solve the above defined Boolean optimization problem. Dealing with this question, it turned out to be useful to replace the objective by a preorder. The problem then is to determine a maximum of a given preorder on $\mathbf{B}^n$ in $S \subseteq \mathbf{B}^n$. Concerning some partial orders on $\mathbf{B}^n$ problems are characterized for which the optimal solution does not depend on the special choice of the objective. Assumptions with regard to S are closely related to matroid theory; in view of the preorder a certain monotonicity condition is important. The dual greedy algorithm and a modified form of it leads us to the definition of dual partial orders. Herewith it is possible to characterize those $S \subseteq \mathbf{B}^n$ for which the greedy algorithm and its dual determine the same vector.

1. Introduction

For $\mathbf{B} = \{0, 1\}$ and an ordered set $(H, \leq)$ consider the Boolean optimization problem (BOP)

$$\text{(P1)} \qquad \max_{x \in S} f(x)$$

with a function $f : \mathbf{B}^n \to H$ and a subset S of $\mathbf{B}^n$. During the last years, is appeared that it is very unlikely to expect "good" algorithms — in the sense of Edmonds' polynomial bounded algorithms — for such arbitrary generic problems. On the other hand there are "good" algorithms for special problems, for example the greedy algorithm. In this paper we describe a class of problems which can be solved by the application of this algorithm and/or its dual. The greedy algorithm has been treated before by Kruskal [8], Edmonds [5], Gale [6], Dunstan and Welsh [4], Magazine, Nemhauser, and Trotter [10], and others.

2. Some binary relations on $\mathbf{B}^n$ (combinatorial structure)

Let us introduce some notations for binary relations R on $\mathbf{B}^n$. R is called a partial preorder, if it is reflexive and transitive; if the adjective "partial" is omitted, then either $x R y$ or $y R x$ must hold; if the prefix "pre-" is omitted, then R is antisymmetric.

Between the subsets of $N = \{1, 2, \ldots, n\}$ and the vectors of $\mathbf{B}^n$ there is an one-to-one correspondence; every vector x is the incidence vector of its support set $T(x) := \{i \in N \mid x_i = 1\}$. The *subvector relation*

$$(2.1) \qquad y \sqsubseteq x :\iff T(y) \subseteq T(x)$$

is thus a partial order on $\mathbf{B}^n$. Corresponding to the union of the support sets we define an *addition*

$$(2.2) \qquad (x+y)_i := \begin{cases} 1 & \text{if } x_i = 1 \text{ or } y_i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

for all $i \in N$. $(\mathbf{B}^n, +)$ is a semigroup. For an arbitrary nonempty subset S of $\mathbf{B}^n$ let be

$$\bar{S} := \{y \in \mathbf{B}^n \mid \exists x \in S : T(y) \subseteq T(x)\}.$$

We can compute the lexicographical maximum $x(S)$ by application of the *greedy algorithm* (A)

(1) $x := 0$; $j := 1$;
(2) if $x + e_j \in \bar{S}$, set $x := x + e_j$;
(3) if $j = n$, stop;
otherwise set $j := j + 1$; return to (2).

(2.3) Definition. Let $x, y \in \mathbf{B}^n$. Then $x \sqsubseteq^i y$ ($x \sqsubseteq^b y$) if there exists an injective (bijective) function $\varphi : T(x) \to T(y)$ such that

$$\varphi(j) \leq j, \quad \forall j \in T(x).$$

The two binary relations defined in (2.3) are partial orders and have the following properties:

(2.4) Proposition. *Let $x, y \in \mathbf{B}^n$. Then*
(1) $x \sqsubseteq^b y \implies x \sqsubseteq^i y$;
(2) $x \sqsubseteq y \implies x \sqsubseteq^i y$;
(3) $x \sqsubseteq^i y \implies x \preceq y$.

Proof. (1) and (2) follow immediately by definition.

(3) Let $\varphi : T(x) \to T(y)$ be injective with $\varphi(j) \leq j$. Suppose $y \prec x$ and let k be the minimal element in $T(x) \setminus T(y)$. Then $\varphi[T_k(x)] \subseteq T_k(y)$ but $|T_k(x)| = |T_k(y)| + 1$ with $T_k(x) := \{i \in T(x), i \leq k\}$. This is a contradiction to the injectivity of φ.

(2.5) Definition. Let R be a binary relation on $\mathbf{B}^n$ and $S \subseteq \mathbf{B}^n$. Then $x \in \mathbf{B}^n$ is called a *maximum (minimum) of S with regard to R* if
(1) $x \in S$,
(2) $y R x$ ($x R y$), $\forall y \in S$.

The set of all such maxima is denoted by $\max_R(S)$. If R is a partial order, then the existence of a maximum implies its uniqueness. Furthermore, (2.4) yields

(2.6) Corollary. *Let $R \in \{\sqsubseteq^n, \sqsubseteq^*, \subseteq\}$. If $x \in \mathbf{B}^n$ is the maximum of S with regard to R, then $x = x(S)$.*

The subsets of $\mathbf{B}^n$ which have a maximum with regard to the partial order $\sqsubseteq^n$ (respectively $\sqsubseteq^*$) are closely related to matroids. Let be $T(S) := \{T(x) \mid x \in S\}$.

(2.7) Definition (*bases*). Let $B \subseteq \mathbf{B}^n$. Then $M = M(N, T(B))$ is a matroid, if

(2.7.1) for all $I, J \in T(B)$, $I \not\subseteq J \wedge J \not\subseteq I$, provided $I \neq J$,

(2.7.2) for all $I, J \in T(B)$, if $j \in J$, then there exists an element $i \in I$ such that $(J \setminus \{j\}) \cup \{i\} \in T(B)$.

The elements of $T(B)$ are called the bases of the matroid M. All bases have equal cardinality.

(2.8) Definition (*independent sets*). Let $S \subseteq \mathbf{B}^n$. Then $M = M(N, T(S))$ is a matroid, if

(2.8.1) for all $J \in T(S)$, $I \subseteq J \Rightarrow I \in T(S)$;

(2.8.2) for all $I, J \in T(S)$, if $|I| < |J|$, then there exists an element $j \in J$ such that $I \cup \{j\} \in T(S)$.

The elements of $T(S)$ are called the independent sets of the matroid M. (2.7) and (2.8) are equivalent definitions as known from matroid theory. The bases are the maximal independent sets and vice versa a subset of a base is an independent set.

(2.9) Definition. Denote the set of all permutations $\pi : N \to N$ by P_n. Then for $\pi \in P_n$, $\hat{\pi} : \mathbf{B}^n \to \mathbf{B}^n$ is the bijective function defined by $[\hat{\pi}(x)]_{\pi(i)} := x_i$ for $i \in N$.

The relationships between matroids and the partial orders defined in (2.3) are given by a theorem corresponding closely to the results of Gale [6].

(2.10) Theorem. *Let $B \subseteq \mathbf{B}^n$ such that* (2.7.1) *holds for $T(B)$. Equivalent statements are*

(2.10.1) *$M = M(N, T(B))$ is a matroid.*

(2.10.2) *for all $\pi \in P_n$ there exists the maximum of $\hat{\pi}(B)$ with regard to $\sqsubseteq^n$.*

Before proving (2.10) we state an equivalent theorem which can be verified by (2.7), (2.8), and (2.3).

(2.11) Theorem. *Let $S \subseteq \mathbf{B}^n$ such that* (2.8.1) *holds for $T(S)$. Equivalent statements are*

(2.11.1) *$M = M(N, T(S))$ is a matroid,*

(2.11.2) *for all $\pi \in P_n$ there exists the maximum of $\hat{\pi}(S)$ with regard to $\sqsubseteq^*$.*

Thus matroids yield special examples for subsets of $\mathbf{B}^n$ which have a maximum with regard to $\sqsubseteq^{\pi}$ (respectively $\sqsubseteq^{\hat{\pi}}$).

Proof of (2.10) ($\Longrightarrow$) If $M = M(N, T(B))$ is a matroid, then $\pi M = M(N, T(\hat{\pi}(B)))$ is a matroid for all $\pi \in \mathbf{P}_n$. Thus we have to show only the existence of the maximum of B with regard to $\sqsubseteq^{i}$. Let $x = x(B)$ and choose $y \in B$. Let

$$T(x) = \{i_1, i_2, \ldots, i_r\}, \quad i_1 < i_2 < \cdots < i_r,$$

$$T(y) = \{j_1, j_2, \ldots, j_r\}, \quad j_1 < j_2 < \cdots < j_r.$$

If $i_k \geq j_k$ holds for all $1 \leq k \leq r$ then $y \preceq x$. Otherwise suppose $m := \min\{k : i_k < j_k\}$. Since $T(B)$ is the set of bases of a matroid, it follows from (2.7) and (2.8) that there exist

$$j \in \{j_1, j_2, \ldots, j_m\},$$

$$i'_{m+1}, \ldots, i'_r \in \{i_m, \ldots, i_r\}$$

such that $\{i_1, \ldots, i_{m-1}, j, i'_{m+1}, \ldots, i'_r\} \in T(B)$. Let x' be the incidence vector of this set. Then by $x \underset{\neq}{\prec} x'$ we have a contradiction.

($\Longleftarrow$) We have to verify (2.7.2). Let $u, v \in B$. The case $|T(u) \setminus T(v)| \leq 1$ is very easy. Otherwise suppose the existence of $i \in T(u) \setminus T(v)$ such that

$$(T(u) \setminus \{i\}) \cup \{j\} \notin T(B) \qquad \forall j \in T(v) \setminus T(u).$$

Now we choose $\pi \in \mathbf{P}_n$ such that elementwise we have

(2.12) $\qquad \pi[T(u) \cap T(v)] < \pi[(T(u) \setminus T(v)) \setminus \{i\}] < \pi[T(v) \setminus T(u)] < \pi[\text{rest}].$

Let be $x = x(\hat{\pi}(B))$ which is the maximum of $\hat{\pi}(B)$ with regard to $\sqsubseteq^{i}$ by (2.6). Clearly

$$|T(v)| = |T(u)| = |T(x)|,$$

$$\hat{\pi}(v) \underset{\neq}{\preceq} \hat{\pi}(u) \preceq x =: \hat{\pi}(\bar{x}).$$

By virtue of (2.12) it follows that there exists an element $j \in T(v) \setminus T(u)$ such that

$$T(\bar{x}) = (T(u) \setminus \{i\}) \cup \{j\}$$

and hence a contradiction.

(2.13) Lemma. *Let $a \in \mathbf{B}^n$. Then $B := \{x \in \mathbf{B}^n \mid x \preceq a\}$ defines a matroid $M = M(N, T(B))$.*

Proof. We have to verify (2.7.1) and (2.7.2). If $|B| = 1$, this is trivial. Otherwise let be $x, y \in B$ and $x \neq y$. From the definition of B it follows that

$$|T(x)| = |T(a)| = |T(y)|$$

which implies (2.7.1). Let be $i \in T(x) \setminus T(y)$. If there exists $j \in T(y) \setminus T(x)$ such that $i \leq j$, then

$$T(\bar{x}) := (T(x) \setminus \{i\}) \cup \{j\} \in T(B)$$

for $\bar{x} \leq^{*} x$. Otherwise define

$$j := \max(T(y) \setminus T(x)).$$

Then $j < i$ and $|T_k(x)| < |T_k(y)|$ for all $j \leq k < i$, and therefore for $T(\bar{x}) := (T(x) \setminus \{i\}) \cup \{j\}$,

$$|T_k(\bar{x})| = |T_k(x)| + 1 \leq |T_k(y)| \text{ for all } j \leq k < i.$$

As $x \leq^{*} a$ and $y \leq^{*} a$ this implies

$$|T_k(\bar{x})| \leq |T_k(a)| \quad \text{for all } 1 \leq k \leq n.$$

This is equivalent to $\bar{x} \leq^{*} a$. By $|T(\bar{x})| = |T(a)|$ it follows that $\bar{x} \leq^{*} a$, i.e. $\bar{x} \in B$.

The set B in the preceding lemma is a special case of a *regular* set with regard to a partial order.

(2.14) Definition. Let be $S \subseteq \mathbf{B}^n$ and R a partial order in $\mathbf{B}^n$. Then S is called *regular with regard to* R, if for all $x \in S$

$$y\,R\,x \implies y \in S.$$

In this sense the set $\bar{S}$ is a regular set with regard to $\leq_{*}$. Regular sets with regard to $\leq^{*}$ have been considered in the literature by Hammer, Johnson and Peled [7] and Wolsey [11]. (2.10) and (2.13) imply

(2.15) Theorem. *Let be $B \subseteq \mathbf{B}^n$ a regular set with regard to $\leq^{*}$. The following statements are equivalent:*

(1) *$M = M(N, T(B))$ is a matroid,*

(2) *there exists the maximum of B with regard to $\leq^{*}$.*

In [11, Theorem 5.5] Wolsey proves an equivalent theorem, which is in our notation

(2.16) Theorem. *Let be $S \subseteq \mathbf{B}^n$ a regular set with regard to $\leq^{*}$. The following statements are equivalent:*

(1) *$M = M(N, T(S))$ is a matroid,*

(2) *there exists the maximum of S with regard to $\leq^{*}$.*

The class of regular sets with regard to $\leq$ and the class of sets which yield a matroid thus only overlap in a very special case.

3. Reformulation of the BOP (algebraic structure)

A given objective function f induces a preorder on $\mathbf{B}^n$ by

(3.1) $\quad x \precsim y :\Longleftrightarrow f(x) \preceq f(y).$

An equivalent formulation of (P1) is therefore

(P2) $\quad$ **Determine** $\quad x \in \max_{\precsim} (S).$

Without loss of generality we only consider functions with the property

(3.2) $\quad f(e_n) \preceq f(e_{n-1}) \preceq \cdots \preceq f(e_1)$

or in view of (P2)

(3.2') $\quad e_n \precsim e_{n-1} \precsim \cdots \precsim e_1.$

Apparently this coincides with the lexicographical ordering of the unit vectors.

(3.3) Definition. Let $\precsim$ be a preorder on $\mathbf{B}^n$. $(\mathbf{B}^n, +, \precsim)$ is called a *preordered semigroup* if for all $x, y \in \mathbf{B}^n$ and all $e_i \not\leq x \vee y$ (M) holds:

(M) $\quad x \precsim y \Longrightarrow x + e_i \precsim y + e_i.$

The monotonicity property (M) is a restriction of the choice of f (resp. of the preorder). In view of (P1) there is an important example of a preordered semigroup.

(3.4) Example. Let $(H, *, \preceq)$ be an ordered semigroup, that is

(3.4.1) $(H, \preceq)$ is an ordered set,
(3.4.2) $(H, *)$ is a semigroup,
(3.4.3) $a \preceq b \Longrightarrow a * c \preceq b * c \quad \forall a, b, c \in H,$

and define with $c_1, c_2, \ldots, c_n \in H$ a special objective

(3.4.4) $\quad f(x) := \sum^{*}_{x_i = 1} c_i$

Then f is well-defined and induces a preorder $\precsim$ such that $(\mathbf{B}^n, +, \precsim)$ is a preordered semigroup.

Proof. Let $x \precsim y$ and $e_i \not\leq x \vee y$. Then $x_i = y_i = 0$ and therefore

$$f(x + e_i) = f(x) * c_i \preceq f(y) * c_i = f(y + e_i)$$

which implies $x + e_i \precsim y + e_i$ by definition.

The vectors of $\mathbf{B}^n$ have a canonical representation by unit vectors

(3.5) $$x = \sum_{i=1}^{n} x_i \cdot e_i$$

with $0 \cdot e_i := 0$ and $1 \cdot e_i := e_i$ for $i \in N$.

(3.6) Definition. Let $\lesssim$ be a preorder on $\mathbf{B}^n$, $k := \min\{j \in N \mid e_j \lesssim 0\}$ (if the set is empty put $k := n+1$). Then

$$x^* := \sum_{i<k} x_i \cdot e_i$$

(3.7) Definition. Let $T(x) = \{j_1, j_2, \ldots, j_r\}$ with $j_1 < j_2 < \cdots < j_r$. Then $x^{(i)}$ is given by

$$T(x^{(i)}) = \{j_1, j_2, \ldots, j_{\min(i,r)}\} \quad \text{for } i \in N.$$

The special subvectors defined by (3.6) respectively (3.7) have some useful properties in preordered semigroups. Let be $\|x\| := |T(x)|$.

(3.8) Proposition. *Let $(\mathbf{B}^n, +, \lesssim)$ be a preordered semigroup, $r = \|x\|$. Then*

(3.8.1) $\quad 0 \lesssim x^*$

(3.8.2) $\quad y \sqsubseteq' x \implies y \lesssim x^*$

(3.8.3) $\quad \|y\| = i \wedge y \sqsubseteq' x \implies y \lesssim x^{(i)}$

(3.8.4) there exists $t \in N$ such that $x^{(t)} = x^*$,

$$x^{(1)} \lesssim x^{(2)} \lesssim \cdots \lesssim x^{(t-1)} \lesssim x^*,$$
$$x^{(r)} \lesssim x^{(r-1)} \lesssim \cdots \lesssim x^{(t+1)} \lesssim x^*.$$

(3.8.1)–(3.8.4) follow by the monotonicity property (M) and (3.2'). Let us consider for example (3.8.2). Let φ correspond to the definition of $y \sqsubseteq' x$. Then, by repeated application of (M)

$$y = \sum y_i \cdot e_i \lesssim \sum y_i \cdot e_{\varphi(i)} =: y'$$

for $\varphi(i) \leq i$ implies $e_i \lesssim e_{\varphi(i)}$ by (3.2)'. Analogously

$$y' \lesssim \sum_{\varphi(i)<k} y_i \cdot e_{\varphi(i)} \lesssim \sum_{i<k} x_i \cdot e_i = x^*$$

with k as defined in (3.6).

The application of the greedy algorithm to S yields step by step the sequence

(3.9) $$x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(n)} = x(S)$$

thus by (3.8.4) it is possible to determine $x(S)^*$. An immediate consequence of (3.8.2) and (2.6) is

(3.10) Theorem. *Let $\precsim$ be a preorder on $\mathbb{B}^n$ and $S \subseteq \mathbb{B}^n$. If*

(3.10.1) *$(\mathbb{B}^n, +, \precsim)$ is a preordered semigroup,*

(3.10.2) *there exists the maximum x^+ of S with regard to $\preceq^+$,*

then $y \precsim x^+$, $\forall y \in S$.

The two assumptions describe a class of problems which can be solved by the application of the greedy algorithm. After the determination of x^+ one has to check whether $x^+ \in S$ or not. If $x^+ \in \bar{S} \setminus S$, then it is only an upper bound. If $S = \bar{S}$, then x^+ is a solution of (P2). The following theorem implies by (3.8.4) Theorem (3.10). Let us denote $S_i := \{x \in S \mid \| x \| = i\}$

(3.11) Theorem. *The assumptions of (3.10) yield for all $1 \le i \le |S| := \max\{\| x \| \mid x \in S\}$*

$$y \precsim x^{(i)}, \quad \forall y \in (\bar{S})_i.$$

The theorem follows from (3.8.3) and (2.6). The two theorems refer to different combinatorial structures.

(3.12) Corollary. *Let $B \subseteq \mathbb{B}^n$. If $M = M(N, I(B))$ is a matroid by (2.7) and $\precsim$ is defined by (3.4), then*

$$x(B) \equiv \max_{\precsim}(B).$$

(3.13) Corollary. *Let $S \subseteq \mathbb{B}^n$. If $M = M(N, T(S))$ is a matroid by (2.8) and $\precsim$ is defined by (3.4) then*

$$[x(S)] \equiv \max_{\precsim}(S).$$

The two corollaries follow from (3.11) respectively (3.10) and (2.10) respectively (2.11). We consider the following class of functions in view of (P1):

(3.14) Definition. *Let F denote the set of all functions $f : \mathbb{B}^n \to H$ with*

(3.14.1) *$(H, \preceq)$ is an ordered set,*

(3.14.2) *$f(e_n) \preceq f(e_{n-1}) \preceq \cdots \preceq f(e_1)$,*

(3.14.3) *$(\mathbb{B}^n, +, \precsim)$ is a preordered semigroup with regard to the preorder induced by f.*

(3.15) Corollary. *Let $S \subseteq \mathbb{B}^n$ with $S = \bar{S}$. If (3.10.2) holds, then regardless of the choice of the objective $f \in F$, $[x(S)]^+$ is a solution of the problem $\max_{x \in S} f(x)$.*

This follows from (3.10). Clearly there is an analogous corollary corresponding to (3.11).

(3.16) Corollary. *Let $B \subseteq \mathbb{B}^n$ with $\| y \| = \| B \|$, $\forall y \in B$. If (3.10.2) holds then regardless of the choice of the objective $f \in F$ $x(B)$ is a solution of the problem $\max_{x \in B} f(x)$.*

These corollaries reflect the fact that the greedy algorithm only considers the values of the objective function for the unit vectors.

At the end of Section 2 we introduced regular sets with regard to ⊑'. As shown by (2.16) in this case the assumption (3.10.2) implies that $M = M(N, T(S))$ is a matroid. Results for more general regular sets with regard to $\subseteq$ and ⊑' are given by Hammer, Johnson and Peled in [7]. If the objective "agrees" with the partial order R, that is

$$(3.17) \qquad xRy \implies f(x) \geq f(y), \quad \forall x, y \in \mathbf{B}^n,$$

then

$$(3.18) \qquad \max_{\leq} S \subseteq \max_{\leq} S_R$$

clearly holds for $S \subseteq \mathbf{B}^n$, $S_R := \{x \in \mathbf{B}^n \mid \exists y \in S : xRy\}$.

If distinct vectors in (3.17) imply distinct function values, then equality holds in (3.18). In this case the BOP (P1) is equivalent to

$$(\mathrm{P3}) \qquad \max_{x \in S_R} f(x).$$

As shown in [7] S_R can be described by the restrictions of a covering problem, that means all restrictions are of the form

$$\sum_{j \in J} (1 - x_j) \geq 1, \quad \text{with } J \subseteq N.$$

In the case $R =$ [illegible] a further simplification is possible and developed in [7].

In connection with covering problems the partial order [illegible] has been considered by Bowman and Starr [2]. They present an enumerative algorithm for the problem of maximizing a partial order on $\mathbf{B}^n$ which fulfills (3.2)' and (M) in (3.3). If in this section $\leqq$ denotes only a *partial* preorder, then under the additional assumption to (3.2)'

$$(3.2'') \quad 0 \leqq e_n \quad \text{or} \quad e_1 \leqq 0 \quad \text{or there exists } k \in N \setminus \{1\} \text{ such that } e_k \leqq 0 \leqq e_{k-1}$$

all results hold which refer to $\leqq$.

4. Dual partial orders

(4.1) Definition. Let R be a partial order on $\mathbf{B}^n$. Then the *dual partial order of R* is R', defined by

$$xR'y :\iff \bar{\sigma}(y) R \bar{\sigma}(x)$$

with $\sigma \in P_n$, $\sigma(i) := n - i + 1$ for $i \in N$.

Partial orders and their duals may coincide more or less.

(4.2) Proposition. (1) $x \subseteq' y \iff y \subseteq x$.
(2) x [illegible] $y \iff x$ [illegible] y.

In view of proposition (2.4) the dual partial orders of those partial orders defined by (2.1) and (2.3) have analogous properties.

(4.3) Proposition. (1) $y \subset x \implies x \sqsubseteq^{b'} y$,
(2) $x \sqsubseteq^{b'} y \implies x \sqsubseteq^{l'} y$,
(3) $x \sqsubseteq' y \implies x \preccurlyeq' y$.

In connection with dual partial orders we consider a *modified greedy algorithm* (A')

(1) $x := 0$; $j := n$;
(2) if $x + e_j \in \bar{S}$, set $x := x + e_j$;
(3) if $j = 1$, stop.
otherwise set $j := j - 1$ and return to (2).

The output vector of this algorithm applied to $S \subseteq \mathbf{B}^n$ shall be denoted by $x'(S)$. The application of (A') to $S^c := \{1 - x \mid x \in S\}$ is called *dual greedy algorithm*.

(4.4) Proposition. *$x'(S)$ is the minimum of S with regard to $\preccurlyeq'$.*

Proof. The application of (A') to S is equivalent to the application of (A) to $\bar{\sigma}(S)$. Hence $x'(S) = x(\bar{\sigma}(S))$. (4.4) follows by (4.1).

For an arbitrary set S the four vectors representing the maxima respectively the minima of S with regard to $\preccurlyeq$ respectively to $\preccurlyeq'$ may be pairwise distinct. For example, take $S = \{x, y, u, v\}$ with

$x = (1\,0\,0\,1\,0)$, maximum with regard to $\preccurlyeq$,

$y = (0\,1\,1\,0\,0)$, maximum with regard to $\preccurlyeq'$,

$u = (0\,1\,0\,0\,1)$, minimum with regard to $\preccurlyeq'$,

$v = (0\,0\,1\,1\,0)$, minimum with regard to $\preccurlyeq$.

(4.5) Proposition. *Let $R \in \{\subseteq, \supseteq, \sqsubseteq^b, \sqsubseteq, \sqsubseteq', \preccurlyeq, \preccurlyeq'\}$. Then*

$$x R y \iff (1 - y) R (1 - x).$$

Let us show this for example in the case of $R = \sqsubseteq$. Equivalent to the left side there is

$$|T_k(x)| \leq |T_k(y)|, \quad \forall 1 \leq k \leq n$$

and this is equivalent to

$$|T_k(1 - x)| \geq |T_k(1 - y)|, \quad \forall 1 \leq k \leq n.$$

The rest follows analogously to the first equivalence. An immediate consequence of (4.5) is the next proposition.

(4.6) Proposition. *$1 - x'(S^*)$ is the maximum of S with regard to $\preccurlyeq'$. $1 - x(S^*)$ is the minimum of S with regard to $\preccurlyeq$.*

The lexicographical maximum or minimum of S as well as the dual lexicographical maximum or minimum of S can be computed by the application of (A) or (A') to S or S^*.

(4.7) Theorem. *Let $B \subseteq \mathbb{B}^n$. The following statements are equivalent:*

(4.7.1) *there exists the maximum $x \in B$ with regard to $\sqsubseteq^b$*

(4.7.2) *there exists the common maximum $x \in B$ with regard to $\sqsubseteq^l$ and $\sqsubseteq^{l'}$.*

An implication of (4.7.1) or (4.7.2) is

(4.7.3) $\quad x(S) = 1 - x'(S^*)$.

Proof. (4.7.1) implies (4.7.2) by (2.4) and (4.3). Reversely, if $y \sqsubseteq^l x$ and $y \sqsubseteq^{l'} x$, then follows by definition $|x| \leqslant |y|' \leqslant |x|$ and therefore $y \sqsubseteq^b x$.

(4.7.2) implies (4.7.3) by (2.6), (4.3) and (4.6).

If (4.7.1) or (4.7.2) hold, the dual greedy algorithm yields the complement of the lexicographical maximum of S. This may be important in view of problem (P1). The crucial point in the application of the greedy algorithm is the test whether $x \in \bar{S}$ or not. If $x \in \overline{(S^*)}$ is easier to check, then one will prefer the dual greedy algorithm.

5. Remarks

The combinatorial structure of problems for which the greedy algorithm is valid is closely related to matroids. The corresponding algorithm for the intersection of two matroids, namely the weighted intersection algorithm of Lawler [9], has not yet been considered in this way, but similar studies have been published by Burkard, Hahn, and Zimmermann [3] as well as Burkard [2] about the assignment problem which is a special example of the intersection of two matroids. Already in this special case it turned out that similar results as in (3.15) cannot be attained, yet an algorithm is stated in [3] which solves the assignment problem with generalized objectives.

References

[1] V.J. Bowman, and J.H. Starr, Set covering by ordinal cuts II, Management Sciences Research Reports No 321/322, 1973 Carnegie-Mellon University Pittsburgh, Pennsylvania.

[2] R. Burkard, Kombinatorische Optimierung in Halbgruppen in: R. Bulirsch, W. Oettli, J. Stoer, eds., Optimization and Optimal Control, Lecture notes in mathematics, 477 (Springer, Berlin, 1975) pp. 1–17.

[3] R. Burkard, W. Hahn and U. Zimmermann, An algebraic approach to assignment problems, Report 1974-1, Mathematisches Institut der Universität Köln 1974.

[4] F.D.J. Dunstan and D.J.A. Welsh, A greedy algorithm for solving a certain class of linear programmes, Math. Programming 5 (1973) 338–353.

[5] J. Edmonds, Matroids and the greedy algorithm, Math. Programming 1 (1971) 127–136.

[6] D. Gale, Optimal assignments in an ordered set: an application of matroid theory, J. Comb. Theory 4 (1968) 176–180.

[7] P.L. Hammer, E.L. Johnson and U.N. Peled, Regular 0-1-programs, Research Report CORR 73-18, University of Waterloo.

[8] J.B. Kruskal, On the shortest spanning subtree of a graph and the travelling salesman problem, Proc. Am. Math. Soc. 7 (1956) 48–50.

[9] E.L. Lawler, Matroid intersection algorithms, Math. Programming 9 (1975) 31–56.

[10] M.J. Magazine, G.L. Nemhauser and L.E. Trotter, When the greedy solution solves a class of knapsack problems, MRC Technical Summary Report No. 1421 (1974), Mathematics Research Center University of Wisconsin, Madison, Wisconsin, U.S.A.

[11] L.A. Wolsey, Faces for a linear inequality in 0-1-Variables, Math. Programming 8 (1975) 165–178.

Annals of Discrete Mathematics 1 (1977) 551-568

INTEGER LINEAR PROGRAMMING WITH MULTIPLE OBJECTIVES*

Stanley ZIONTS
School of Management, State University of New York at Buffalo, Buffalo, N.Y., U.S.A.

Although it may seem counterintuitive, a method for solving multiple criteria integer linear programming problems is not an obvious extension of methods that solve multiple criteria linear programming problems. The main difficulty is illustrated by means of an example. Then a way of extending the Zionts–Wallenius algorithm [6] for solving integer problems is given, and two types of algorithms for extending it are briefly presented. An example is presented for one of the two types. Computational considerations are also discussed.

1. Introduction

In [6] a method was presented for solving multiple criteria linear programming problems. Because integer programming is a generalization of linear programming in that a subset of variables may be required to take on integer values, it is reasonable to ask if multicriteria integer problems can be solved by an obvious extension to the method: solving the multicriteria linear programming problem using that method and then using the associated multipliers to solve the integer problem. In general, unfortunately, such a procedure is not valid. Assuming that the implicit utility function of the decision maker is a linear additive function of objectives, the general idea can be modified into a workable algorithm for solving mixed or all integer programming problems involving multiple objectives.

Numerous approaches to various problems involving multiple objective functions have been proposed. B. Roy [3] discusses a number of them. He also develops a typology of methods [3, p. 240]:

"1. aggregation of multiple objective functions in a single function defining a complete preference order;
2. progressive definition of preferences together with exploration of the feasible set;
3. definition of a partial order stronger than the product of the n complete orders associated with the n objective functions;
4. maximum reduction of uncertainty and incomparability."

To put things into perspective, the approach of [6] is a combination of 1 and 2 in that an aggregation of the functions is accomplished by an interactive process in

* An earlier version of this paper has also been issued as Working Paper 75-32 of the European Institute for Advanced Studies in Management in Brussels.

which preferences are expressed. The use of multiple criteria in an integer framework has been mentioned in [6] and more recently in [1] and [4].

The plan of this paper is to first indicate why noninteger methods cannot be extended in an obvious way to solve multiple criteria integer problems. Then two extensions of the method of [6] for solving integer problems are developed, an example is solved, and some considerations for implementation are given. In an appendix the method of [6] is briefly overviewed.

2. Some considerations for solving multiple criteria integer problems

The problem to be considered is a mixed integer linear programming problem. Let the decision variables be a vector x of appropriate order where some or all of the variables are required to take on integer values. Denote the set of integer variables as J. The constraint set is then

$$\begin{aligned} &Ax = b \\ &x \geq 0 \qquad\qquad (1) \\ &x_j,\ j \in J \text{ integer,} \end{aligned}$$

where A and b are, respectively, a matrix and vector of appropriate order. In addition we have a matrix of objective functions C where row i of C gives the ith objective C_i. Each objective of u is to be maximized and we may thus write

$$Iu - Cx \leq 0. \qquad (2)$$

The formulation (1)–(2) is the most general formulation of the multiple criteria integer programming problem if one grants that any nonlinearities are already represented in the constraints (1) using piecewise linearizations and integer variables as necessary. If we accept that the implicit utility function is a linear function (as was done originally in [6]) of the objectives u, we may therefore say that our objective is to maximize λu where λ is an unknown vector of appropriate order. Were λ known, the problem of maximizing λu subject to (1) and (2) would be an ordinary integer programming problem. Such a problem could be solved using any method for solving integer linear programming problems. The problem is that λ is not known.

In an earlier paper [6] Wallenius and I developed a method for solving linear programming problems having multiple objectives. That method is briefly summarized in the appendix. The method has been extensively tested and seems to work in practice. A natural extension of that method would appear to be an extension for solving problems involving integer variables:

1. Solve the continuous multiple criteria problem according to the method of [6];

2. Using the multipliers obtained in step 1, solve the associated integer linear programming problem.

Unfortunately as the following simple example shows, that extension does not necessarily work.

Given the constraints:

$$x_1 + \tfrac{1}{3}x_2 \leq 3\tfrac{1}{8}$$

$$\tfrac{1}{3}x_1 + x_2 \leq 3\tfrac{1}{8}$$

$$x_1, x_2 \geq 0 \text{ and integer}$$

with objectives $u_1 = x_1$, and $u_2 = x_2$ then provided that the true multipliers λ_1 and λ_2 (>0) satisfy the following relationships

$$\lambda_1 \geq \tfrac{1}{3}\lambda_2$$

$$\lambda_1 \leq 3\lambda_2$$

then the continuous solution $x_1 = 2.34$, $x_2 = 2.34$ is optimal. However, for this problem there are three optimal integer solutions corresponding to the same continuous optimum depending on the true weights:

If $3\lambda_2 \geq \lambda_1 \geq 2\lambda_2$, then $x_1 = 3$, $x_2 = 0$ is optimal;
If $2\lambda_2 \geq \lambda_1 \geq 0.5\lambda_2$, then $x_1 = x_2 = 2$ is optimal;
If $0.5\lambda_2 \geq \lambda_1 \geq \tfrac{1}{3}\lambda_2$, then $x_1 = 0$, $x_2 = 3$ is optimal.

The example could readily be made more complicated, but it serves to show that further precision may be required in the specification of the multipliers than only to identify the multiplier valid at a noninteger optimal solution. (Further precision is not always required; change the constraint value of the problem from 3.125 to 2.99.)

3. Adapting the Zionts–Wallenius method for solving integer programming problems

To further specify the multipliers λ to find the optimal integer solution, it is necessary to ask additional questions of the decision maker. There are numerous ways in which this may be done, and we shall explore two of them. Both of these proposals represent untested procedures.

3.1. A branch and bound approach

We first consider branch and bound algorithms. The multiple criteria method can be altered to work in a branch and bound integer framework. To do this we first present a flow chart of a simple branch-and-bound algorithm, [5, p. 416] in Fig. 1. As usual, $[y]$ is the largest integer not exceeding y. The idea is to solve a sequence of linear programming problems thereby implicitly enumerating all of the possible integer solutions. The best one found is optimal. The procedure of Fig. 1 cannot be

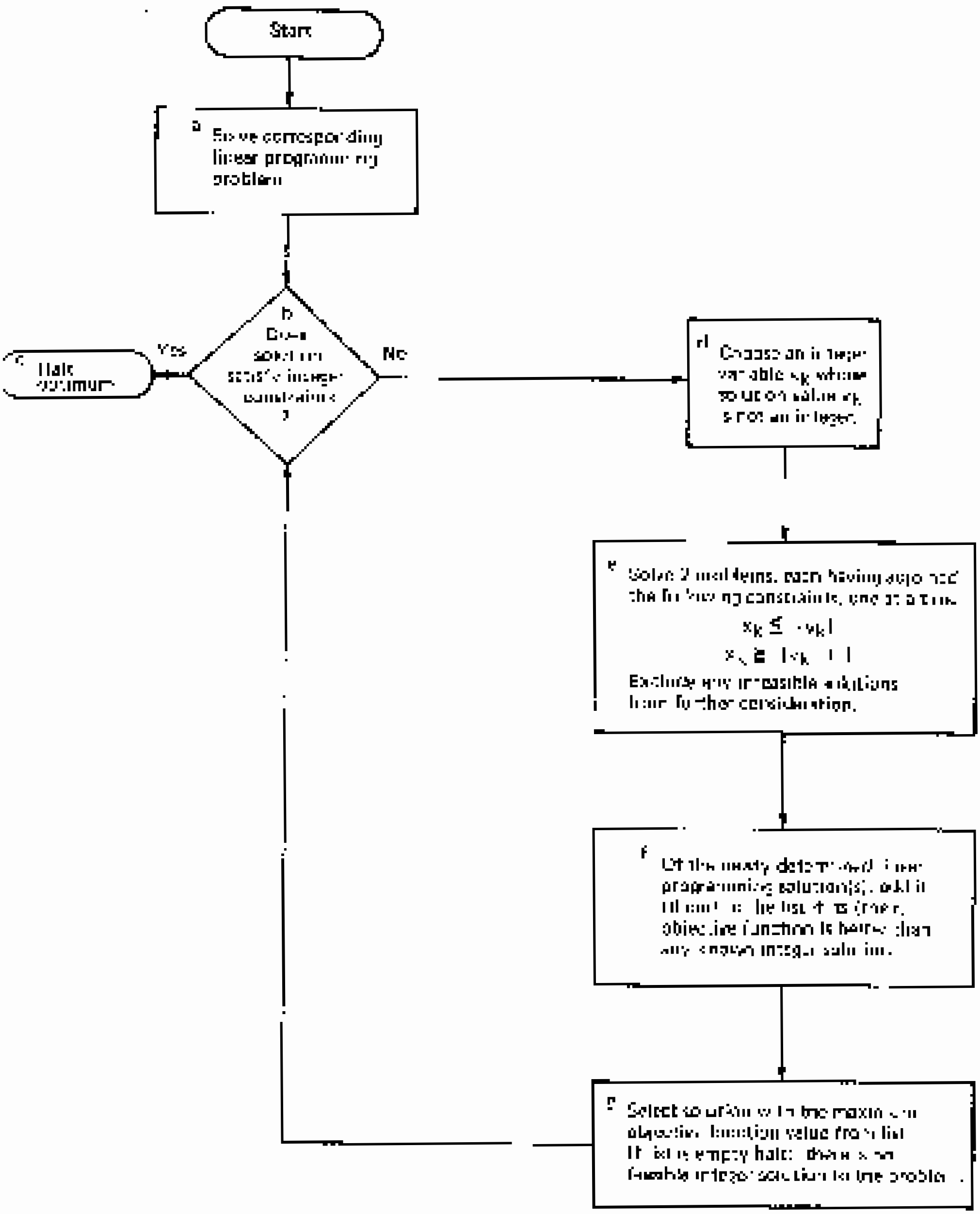


FIG. 2. Flow Chart of a Sample Branch and Bound Algorithm Taken from [5, page 416].

used directly, but must be modified. The modifications which are to be made are based on the following theorem.

Theorem. *A solution can be excluded from further consideration (not added to the list) provided the following two conditions hold:*

(1) the decision maker prefers an integer solution to it,

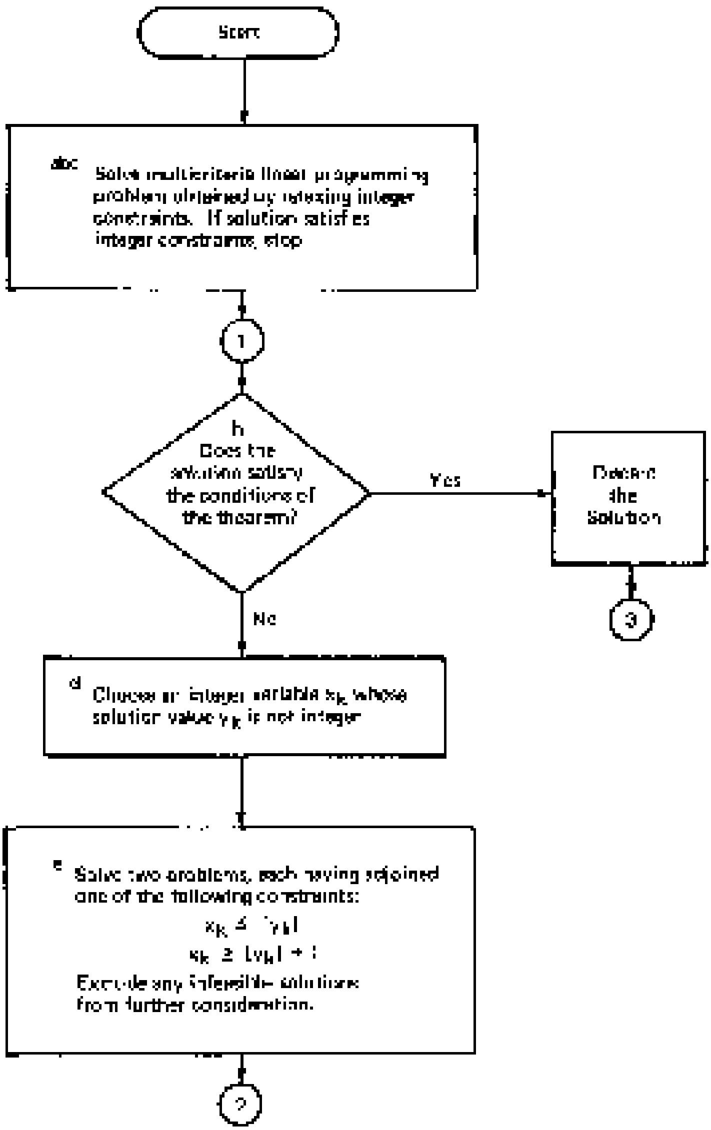


Fig. 2.
Flow Chart of a Branch and Bound Multicriteria
Integer Linear Programming Method

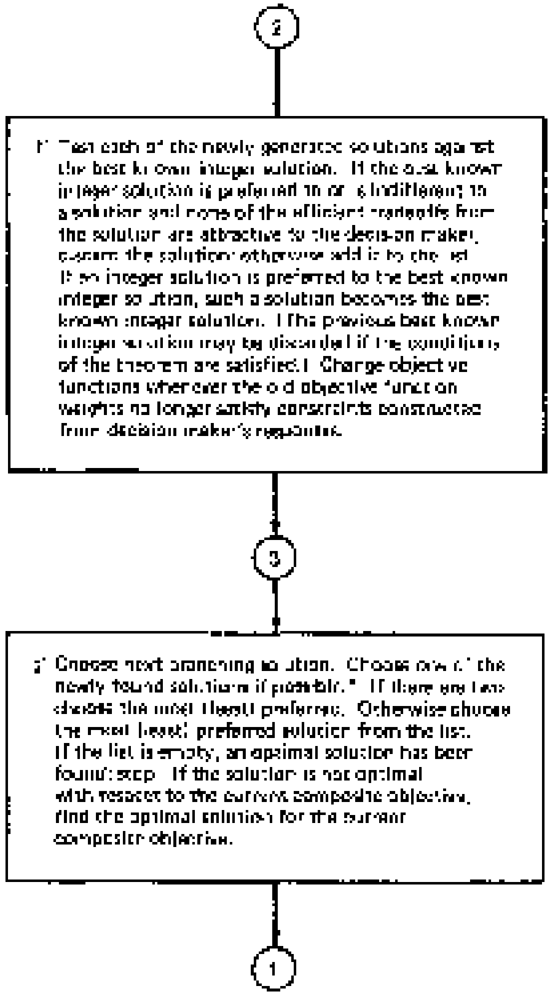


Fig. 2 (Continued)

* A depth first strategy has been adopted. It is an option that may or may not be desirable.

(2) *all efficient tradeoff questions associated with the solution are viewed negatively or with indifference.*

Proof. As shown by the decision-maker's preference the known integer solution has a greater objective function value than the solution in question. Further, since no continuous neighbor is preferred to the solution, any further restricted solution will have a lower objective function than the solution in question and therefore the integer solution.

We were tempted to weaken the second condition of the theorem to a comparison between the known integer solution and the efficient adjacent extreme point solutions of the solution in question by using a slight alteration to our method proposed by Fandel and Wilhelm [2]. Unfortunately, such a change is not valid here.

The question of preference is first checked by comparing the preference relationship with previously expressed preferences (derived from responses) to see whether or not the preferences can be deduced. If that is the case the preference is known; if not a question is posed to the decision maker, and the responses further restrict the multiplier space. Whenever a new set of multipliers is found they are to be substituted for the old set. An algorithm based on the above presentation is given in Fig. 2, and an example will be solved using it.

The letter references correspond in the two figures. Substantial changes have occurred in blocks f and g. Where "most (least) preferred" are indicated are option-points. We have chosen the one not in parentheses, arbitrarily, but not because we have evidence it is superior. Many other options are possible, such as the use of penalty methods in choosing the branching variable, etc., but we have generally ignored such considerations in this paper.

We now present an example, the example presented in Section 2. We use the algorithm of Fig. 2 assuming that the true weights are $\lambda_1 = 0.7$, $\lambda_2 = 0.3$, but that the weights chosen at the continuous optimum are $\lambda_1 = 0.3$, $\lambda_2 = 0.7$. The tree of solutions is given in Fig. 3, and the number in each block indicates the solution number — the order in which each solution is found. (The shaded region is what also would have been generated if every branch had to be terminated either in an integer solution or an infeasible solution without terminating any branches otherwise.) (For this problem no solution had to be re-solved.)

Table 1 is the optimal continuous solution, where x_3 and x_4 are the slack variables. (The identity matrix has been omitted.)

Table 1

		x_3	x_4
x_1	2.34	1.125	0.375
x_2	2.34	−0.375	1.125
u_1	2.34	1.125	0.375
u_2	2.34	−0.375	1.125

The questions to Table 1 are both efficient (this is not demonstrated) and the two questions are found in the last two rows of the table: Are you willing (for variable x_3) to decrease u_1 by 1.125 units and increase u_2 by 0.375 units? A simulated response is obtained by using the true weights. Here we compute $-1.125(.7) + 0.375(.3)$. Since the sum is negative, the simulated response is no. Are you willing

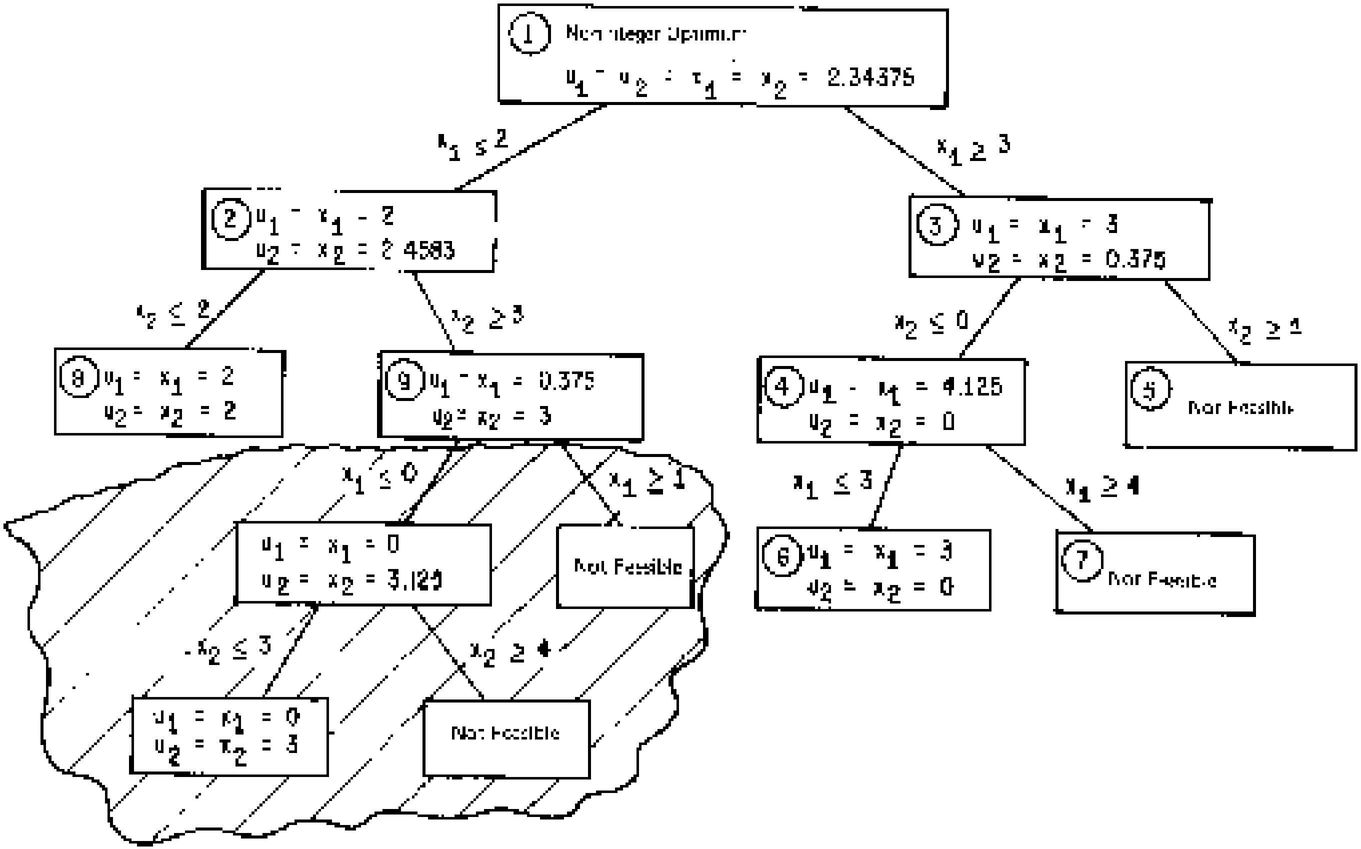


FIG. 5
The Branch and Bound Solution
to the Example Problem

(for variable x_4) to increase u_1 by 0.375 units by having u_2 decrease by 1.125 units? (Simulated response: no). The negative responses confirm the optimality of the solution to Table 1. The constraints are then

$$\lambda_1 \geq \tfrac{1}{3}\lambda_2$$

$$\lambda_1 \leq 3\lambda_2.$$

By using $\lambda_1 + \lambda_2 = 1$, we have on eliminating λ_2:

$$0.25 \leq \lambda_1 \leq 0.75.$$

As indicated above we use $\lambda_1 = 0.5$ (noting that the true value is $\lambda_1 = 0.7$). Solving the two linear programming problems by branching on x_1 from the noninteger optimum we have solutions 2 and 3. Which is preferred is not obvious and we illustrate the test. Solution 3 has a utility of $3\lambda_1 + 0.375\lambda_2$. Solution 2 has a utility of $2\lambda_1 + 2.458\lambda_2$. The difference between the utility of solution 3 and that of solution 2 is

$$\lambda_1 - 2.0833\lambda_2 \overset{?}{\gtrless} 0.$$

On using $\lambda_2 = 1 - \lambda_1$ we have

$$3.0833\lambda_1 - 2.0833 \overset{?}{\gtrless} 0$$

Because $0.25 \leq \lambda_1 \leq 0.75$, the term can be either positive or negative (in general two small linear programming problems must be partially solved to know this); hence a question is asked. Using a simulated test of preference, as above, the decision maker prefers solution 3, and we have a new constraint,

$$3.0833\lambda_1 - 2.0833 \geq 0 \quad \text{or} \quad \lambda_1 \geq 0.675.$$

Thus we now have $0.675 \leq \lambda_1 \leq 0.75$ so we choose $\lambda_1 = 0.72$. We then branch on solution 3 to find solutions 4 and 5 (not feasible) and then branch on solution 4 to find solutions 6 and 7 (not feasible). To this point there have been no known integer solutions; thus the tests against the best known integer solution have been suppressed. Since solution 6 is integer, it becomes the best known integer solution. Next we choose the only remaining solution on the list, solution 2. As the comparison with solution 6 is not implied, the decision maker is asked which solution he prefers. He prefers solution 2; then the constraint

$$\lambda_1 - 2.4583\lambda_2 \leq 0 \quad \text{or} \quad \lambda_1 \leq 0.711,$$

is added and we have $0.675 \leq \lambda_1 \leq 0.711$ and we choose $\lambda_1 = 0.69$. We next branch on solution 2; this yields solutions 8 and 9. Both solutions may be discarded because the conditions of the theorem are satisfied. (The constraints on the λ's are sufficiently tight that all preferences are implied and no questions need to be asked.) Since there are no other solutions on the list, solution 6 has been found to be optimal. The method of Figure 1 using the correct weights enumerates the same solutions except that solutions 8 and 9 are not enumerated.

3.2. A cutting plane approach

To illustrate another algorithm we also present a dual cutting plane approach. It is a logical extension of any dual cutting plane method with respect to multiple criteria decision making. Let k be a nonnegative integer, a choice variable that specifies the frequency of generating additional questions in the absence of finding an integer solution. The parameter k may be sufficiently large as to be effectively infinite. Then the procedure is the following:

(1) Find the continuous multiple criteria optimum using the method of [6] and set i to 0. Use the associated weights to generate a composite objective function.

(2) Adjoin a cut, increment i by one unit and optimize using the present composite objective function. Denote the solution found as the incumbent.

(3) If the incumbent solution is integer, go to 4. Otherwise, if i is not equal to k, go to 2. If i is equal to k go to 5.

(4) Set i to zero, generate efficient questions (see the appendix for the definition) for the current solution that are consistent with previous responses. If the decision maker finds none of the tradeoffs attractive (or if there are no efficient tradeoffs) stop: the optimal solution has been found. Otherwise, use the responses to find a new composite objective function and perform the iterations necessary to achieve a linear programming optimum. Designate the associated solution as the incumbent solution and go to 3.

(5) Set i to zero, generate efficient questions for the current solution that are consistent with previous responses. Use the decision maker's responses to generate a new composite objective function and perform the iterations (possibly none) necessary to achieve a linear programming optimum. Designate the associated solution as the incumbent solution and go to 3.

That this method is valid follows from the fact that every time an integer solution is found (and so long as k is not infinite, more often), questions are generated and the multipliers may be altered by the procedure. Every time step 4 is utilized the optimality of an efficient integer solution (an efficient extreme point of the convex hull of all feasible integer solutions) is confirmed or denied. If it is confirmed, the optimality has been demonstrated; if it is denied, one extreme point of the convex hull of all feasible integer solutions has been eliminated from consideration. So long as the solution space is closed and bounded, the number of such extreme points is finite. Therefore in such a case the procedure is finite.

The effectiveness of choosing k to be finite is not clear, nor is the effectiveness of the method known. How well this scheme works depends on the power of the cut method employed. Since dual cut methods are not currently used much because they do not work well in practice, it is unlikely that a multiple criteria scheme based on a dual cut will work well.

Although approaches may be developed for other integer algorithms, we shall not develop any additional approaches here.

4. Discussion

The implementation of multiple criteria integer programming in liaison with dual cut methods and with branch and bound methods seems straightforward, although it appears warranted only in conjunction with branch and bound methods. Implementation should not be difficult, and it is felt that the difference between solving integer programming problems with multiple criteria and integer programming problems with a single criterion would be roughly the same as the performance of a multiobjective linear program as compared to a single objective linear program. More questions will be asked in the integer case, and probably more partial solutions will be generated as well, but it seems that the increase will not be considerable. A number of tests which correspond to solving relatively very small linear programming problems must be incorporated as well. The above statements are rather speculative and require further testing. For testing purposes, a computer program of the Zionts–Wallenius method now being prepared by the SIDMAR Corporation working together with the University of Ghent may be extended to the integer case and used. It is designed to be an easily usable and alterable program.

In the noninteger case we were able to relax the assumption of the additive utility function to a general concave utility function. Such a generalization in the integer case seems rather unlikely because a point other than an extreme point solution of the convex polyhedron of feasible integer solutions can be optimal in the general concave case. A simple example of such a model would be the use of a utility function involving a product of objectives. (See Bowman [1], for an example.) In the linear case a neighborhood of feasible solutions would be identified and a point in the neighborhood would be optimal. Unfortunately, the use of such a scheme in the integer case would terminate with an integer solution and a neighborhood which need not contain any other feasible integer solutions.

Appendix. Overview of the Zionts–Wallenius method [6] for solving multiple criteria linear programming problems

Let the problem of concern be

$$\begin{aligned} &Ax = b \\ &x \geq 0 \\ &Iu - Cx = 0. \end{aligned} \tag{A.1}$$

The objective is to maximize λu where $\lambda > 0$ but unknown. The procedure is as follows:

(1) Choose an arbitrary $\lambda > 0$.

(2) Solve the associated linear programming problem (A.1). The solution is an efficient solution. Identify the adjacent efficient extreme points in the space of the

objective functions for which a negative answer by the decision maker is not implied. If there are none, stop; the optimal solution has been found. The marginal rates of change in the objectives from the point to an adjacent point is a tradeoff offer, and the corresponding question is called an efficient question.

(3) Ask the decision maker if he likes or dislikes the tradeoff offered for each efficient question.

(4) Find a set of weights λ consistent with all current and previous responses of the decision maker.

Go to step 2.

References

[1] V.J. Bowman, On the Relationship of the Tchebycheff Norm and the Efficient Frontier of Multiple-Criteria Objectives, Paper presented at the conference on Multiple Criteria Decision Making, May 21–23, 1975, at Jouy-en-Josas, France.

[2] G. Fandel and J. Wilhelm, Towards the Theory of Multiple Criteria, Paper presented at the Conference on Multiple Criteria Decision Making, May 21–23, 1975, at Jouy-en-Josas, France.

[3] B. Roy, Problems and Methods with Multiple Objective Functions, *Math. Programming*, 1, 2 (1971) 239–266.

[4] J.F. Shapiro, Multiple Criteria Public Investment Decision Making by Mixed Integer Programming, Paper presented at the Conference on Multiple Criteria Decision Making, May 21–23, 1975, at Jouy-en-Josas, France.

[5] S. Zionts, *Linear and Integer Programming* (Prentice Hall, Englewood Cliffs, NJ, 1974).

[6] S. Zionts and J. Wallenius, An Interactive Programming Method for Solving the Multiple Criteria Problem, Working Paper 74-10, European Institute for Advanced Studies in Management, Brussels, Revised March 1975, *Management Science*, 22, 6 (1976) 652–663.