TRAVELING SALESPERSON PROBLEMS

Many problems of both practical and theoretical importance deal with the search for an optimal solution as defined by Papadimitriou and Steiflitz (1). Wilde and Beightler (2) have identified three major requirements for an optimization method: (1) determine precisely the problem variables and their interaction, (2) develop a measure of problem effectiveness expressible in terms of the problem variables, and (3) choose those values of the problem variables that yield better solutions.

There exist several classes of problems; Pierre (3) provides mathematical definitions of some of them. Optimization problems are usually divided into two main categories: those with continuous variables and those with discrete (combinatorial) variables. In problems with continuous variables, a set of real numbers or a function that provides a solution is generally looked for. In combinatorial problems, a solution that contains a finite integer set, a permutation set, or a graph is searched for. In optimization designs, only certain feature combinations are possible. This means that the possible solutions are restricted to a subregion. The function may be designed to focus on the subregion. The goal of optimization is to find the solution in this subregion for which the function obtains its smallest value, which is referred to by Torn and Zilinxkas (4) and other researchers as the *global minimum.*

Among optimization problems, the traveling salesperson problem (TSP), classified as an NP-complete problem, has been widely studied. For this problem, no algorithm has been demonstrated to find the optimum solution in polynomial time. In a 20-city TSP, there are roughly 6×10^{16} possible tours [$20!/(2 \times 20)$]. Many other problems have similar complexity, such as computer wiring, wallpaper cutting, and job sequencing. Consequently, the solution of this NP-complete problem has been the subject of a large amount of work. The TSP is one of the most prominent of the unsolved combinatorial optimization problems and the most common basis for comparisons. According to Lawler, et al. (5), the TSP continues to influence the development of optimization concepts and algorithms.

The TSP consists of two sets: a set of cities $V = \{1, \ldots, n\}$ and a set of links A. The links are represented by pairs $(i, j) \in A$, meaning that there is a link between city i and city j. The travel distance between city i and city j is expressed as c_{ij}. The problem is to find a tour starting at any city, visit every city exactly once, and return to the starting city. This tour should take the least total traveled distance.

To formulate this problem, a variable x_{ij} is introduced; where $x_{ij} = 1$ if j immediately follows i on the tour and $x_{ij} = 0$ otherwise. The requirement that each city be entered and left exactly once is stated as

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \qquad \text{for} \quad j \in V \tag{1}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1 \qquad \text{for} \quad j \in V \tag{2}$$

The above constraints are not sufficient to define a tour, since a subtour can satisfy them as well. One way to eliminate subtours is to introduce another constraint. For each subset $U \subset V$, $2 \leq U \leq V - 2$, the constraint is that

$$\sum_{(i,j)\in A: i\in U,\ j\in V\setminus U} x_{ij} \geq 1 \qquad \text{for} \quad j \in V \tag{3}$$

The TSP can be formulated as

$$\min \left\{ \sum_{(i,j)\in A} c_{ij} x_{ij} : x \text{ satisfies Eqs. (1)–(3)} \right\} \tag{4}$$

The number of constraints is nearly $2^{(V)}$.

There are three major types of methods that deal with the TSP, namely, neural networks, heuristic searches, and genetic algorithms. Neural networks potentially offer a powerful tool for solving combinatorial problems. This approach has an embedded parallelism that potentially can be implemented in hardware. The original work can be traced back to Hopfield and Tank (6). The proposed method is to encode the TSP into a two-dimensional neuron array. The approach relies on a fully connected artificial neural network. This type of approach has been presented and evaluated by a number of researchers, among them Aiyer (7), Sanchez-Sinencio and Lau (8), Pretzel et al. (9), and Lin (10).

The other algorithms generally rely on some sort of heuristic or an intelligent guess to find good solutions in reasonable time. The most common techniques range from the simple greedy algorithm to the more complex branch-and-bound search and Lin–Kernighan (11) algorithms. A comprehensive account of such techniques for the TSP can be found in Lawler et al. (5). In general, heuristic algorithms provide a fast approach to a lower bound.

Genetic algorithms mimic natural evolution as a population-based optimization process. They are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured and randomized information exchange to form a search algorithm. In every generation, a new set of artificial chromosomes is created using pieces of the fittest chromosome of the previous generation. Genetic algorithms efficiently exploit historical information to speculate on new search points with expected improved performance (3). These algorithms generally provide good solutions at low speed. Hybrids of heuristic and genetic algorithms for the TSP are presented here, with mathematical models as well as performance results.

EVOLUTIONARY COMPUTATION

Evolutionary computation is the most commonly used term to describe the new computing paradigm that mimics natural evolution. Evolutionary computation comprises three major fields: genetic algorithms, evolutionary strategies, and evolutionary programming. Each evolutionary computation algorithm uses similar operations. Each begins with a population of contending trial solutions for the task at hand. New solutions are created by altering the existing solutions using a set of evolutionary computation operators. An objective measure called *fitness* of the trial solution is used to evaluate the new solution. A selection mechanism determines which solutions to maintain as *parents* (or *seeds*) for the subsequent generation. The differences between the procedures are characterized by the types of admissible alterations on parents to create offspring and the methods employed for selecting new parents; readers are referred to Fogel (12,13) and Srinivas and Patnaik (14). In natural evolution, genes on chromosomes carry the environment-fitting information of the species across generations. In genetic algorithms, such information is represented by a fitness function (competitive selection), and chromosomes provide the link between generations. If a chromosome does not fit well within the current environment, it will become extinct. The adaptive procedure is implemented by means of mutation, inversion, or crossover operators. Below the basic genetic algorithm and its operators are presented.

Genetic Algorithm Approaches

The simplest form of a genetic algorithm includes reproduction and selection according to Goldberg (15). Each offspring's fitness value is used as a criterion to perform the selection of the potential new parents (seeds). The basic procedure of the genetic algorithms is shown below in a programlike format. In this procedure t indicates the current generation, and the number $P(t)$ represents the population at generation t:

```
begin
    $t \leftarrow 0$;
    Initialize $P(t_0)$;
    Evaluate $P(t_0)$;
    while no_termination
        begin
            $t \leftarrow t + 1$;
            Select $P(t)$ from $P(t - 1)$;
        Recombine from $P(t)$
        Evaluate $P(t)$;
    **end**
**end.**
```

Goldberg (15) has identified three basic recombination operators in genetics algorithms, namely crossover, inversion, and mutation. These three operators are described in the following sections.

Crossover Operator. The crossover operator requires two chromosomes. These chromosomes exchange a number of genes. The exchanged genes maintain the same relative position with respect to each other. The steps that are required for a crossover operation are listed below. In this description, the chromosomes are referred as strings:

1. Select two mating strings according to a selection policy.
2. Select two points for each string (this selection is usually random).
3. Swap the piece of string within these two points between the two strings.

Figure 1 illustrates the crossover operation. In this case there are two chromosomes called A and B. Each chromosome has 10 genes (which are numbered from 1 to 10). Although some applications allow a duplication of some genes in the same chromosome, for the TSP in general it is considered that each chromosome has a unique set of genes. Figure 1(a) shows the original two chromosomes A and B; the points 1 and 2 are randomly selected. The resulting new chromosomes are shown in Fig. 1(b).

Inversion Operator. The inversion operator works on a chromosome by reversing the sequence of a substring of genes. The steps to follow are:

1. Select a string.
2. Select two points in the string.
3. Invert the piece of the string between the two points.

Figure 2 shows this operation.

Mutation Operator. The mutation operator requires a chromosome (or string) to operate on. The objective of this operator is to slightly disrupt the current chromosome by inserting a new gene. The steps that are followed in the implementation of this operator are listed below:

1. Select a string.
2. Select a site in the current chromosome.
3. Obtain a new gene from a gene pool.
4. Replace this site with the new gene.

Figure 3 shows the mutation operation.

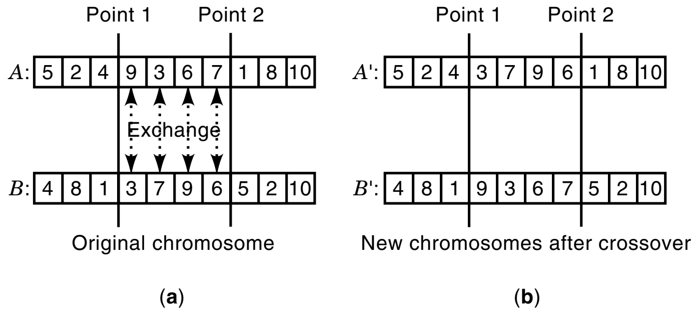


Figure 1. Standard crossover operator.

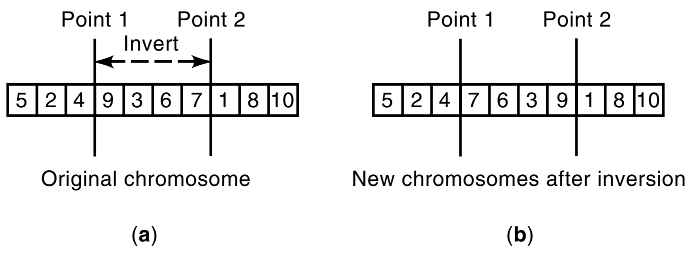


Figure 2. Standard inversion operator.

Genetic Approaches for the Traveling Salesperson Problem

Several genetic approaches to the TSP have been introduced. These approaches include *partially matched crossover* (PMX) by Goldberg (15), *cycle crossover* (CX) by Oliver et al. (16), *order crossover* (OX) by Goldberg (15), *edge recombination* by Whitley et al. (17), *matrix crossover* (MX) by Homaifar et al. (18), and *evolutionary programming* by Fogel (19). These approaches are described in the following sections.

Goldberg Partially Matched Crossover Approach. PMX, introduced by Goldberg (15), has been considered as a way to tackle a blind TSP. The blind TSP approach is not aware of the distance between the cities. The total traversed distance is obtained only at the end of the tour. The application of PMX to the TSP follows the procedure provided below; an example is shown in Fig. 4:

1. Code a tour as a chromosome.
2. Select two chromosomes as parents. PMX proceeds by positionwise exchange. Two crossing sites are picked from a uniform distribution along the string.
3. Map chromosome 2 to chromosome 1. In the example (Fig. 4), $ACHB$ (of chromosome 2) exchange places with $DEFG$ (of chromosome 1).
4. Use a correction procedure to legitimize the TSP path.

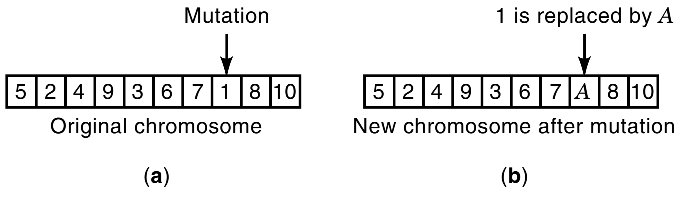


Figure 3. Standard mutation operator.

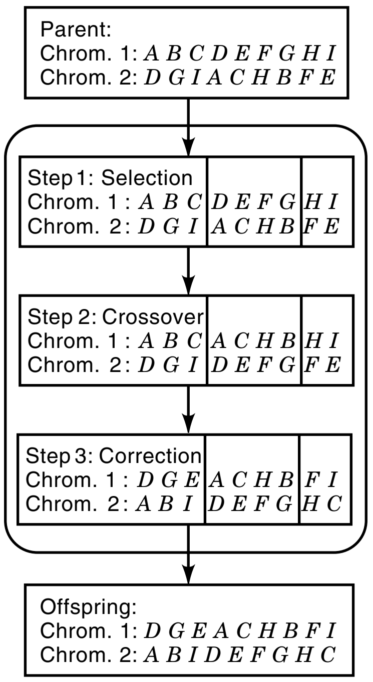

Parent:
Chrom. 1: A B C D E F G H I
Chrom. 2: D G I A C H B F E
Step 1: Selection
Chrom. 1 : A B C | D E F G | H I
Chrom. 2: D G I | A C H B | F E
Step 2: Crossover
Chrom. 1 : A B C | A C H B | H I
Chrom. 2: D G I | D E F G | F E
Step 3: Correction
Chrom. 1 : D G E | A C H B | F I
Chrom. 2: A B I | D E F G | H C
Offspring:
Chrom. 1: D G E A C H B F I
Chrom. 2: A B I D E F G H C

Figure 4. PMX operation.

The benefit of this crossover approach is that each string contains ordering information determined by both its parents. However, the algorithm does not use the distance information.

Oliver Cycle Crossover Approach. The CX approach, introduced by Oliver et al. (16), creates offspring from a pair of parents. Every element of an offspring comes from one of the two parents; the position of each element is identical to that in one of the parents. However, the offspring will be different from both parents. Two rules can be imposed. One is that every position of the offspring must retain a value found in the corresponding position of a parent, and the other is that the offspring must contain a permutation. The operation procedure is explained by means of an example shown in Fig. 5. The steps are as follows:

1. Select two strings as parents as shown in Fig. 5.
2. Randomly select a starting point. Assuming C is selected, the relative position of C in the other chromosome corresponds to I. I is selected in chromosome 1. The relative position of I in chromosome 2 corresponds to E. E is selected from chromosome 1. The relative position of E in the chromosome 2 corresponds to C. Since C has been already selected, a complete cycle is formed.
3. Preserve the selected genes in chromosomes 1 and 2.
4. Switch the selected genes between the two chromosomes.

In a standard crossover operator, a random crossover point, or cut section, is chosen. In the cycle crossover a random parent is chosen for each cycle. The absolute positions of both parents are preserved.

Order Crossover Approach. The OX approach requires two strings. The order crossover starts in a similar way to PMX by selecting a range of genes. Figure 6 shows an example of the use of OX. Below a description of this approach is provided:

1. Randomly select a section of genes as crossover segment.
2. Exchange string 1's segment and string 2's segment.
3. Replace the illegal city (visited twice) with a empty entry.
4. Shift those empty entries together at the opposite ends of each string.
5. Replace the empty entries with the corresponding segment in the other string.

The absolute positions are preserved.

Whitley Edge Recombination. Another operator, introduced by Whitley et al. (17), constructs an offspring tour by exclusively using links present in the two parent structures. On average, these edges should reflect the goodness of the parent structures. There is no random information that might drive the search toward any arbitrary links in the search space. Operators that break links introduce unwanted mutation, which can change the search processing. The edge recombination operator uses an edge map to construct an offspring that inherits as much information as possible from the parent structures. The procedure for this approach is listed below:

1. Assume that there are two strings of chromosomes in the parent, string 1 and string 2, selected to recombine:

   String 1: $A\ B\ C\ D\ E\ F\ G\ H\ I$
   String 2: $D\ G\ I\ A\ C\ H\ B\ F\ E$

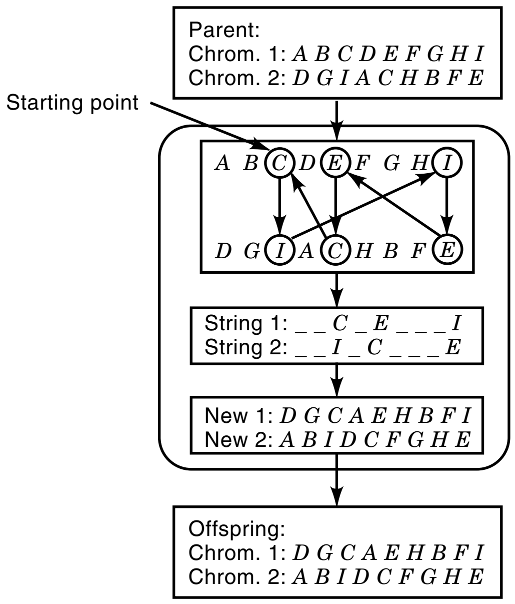
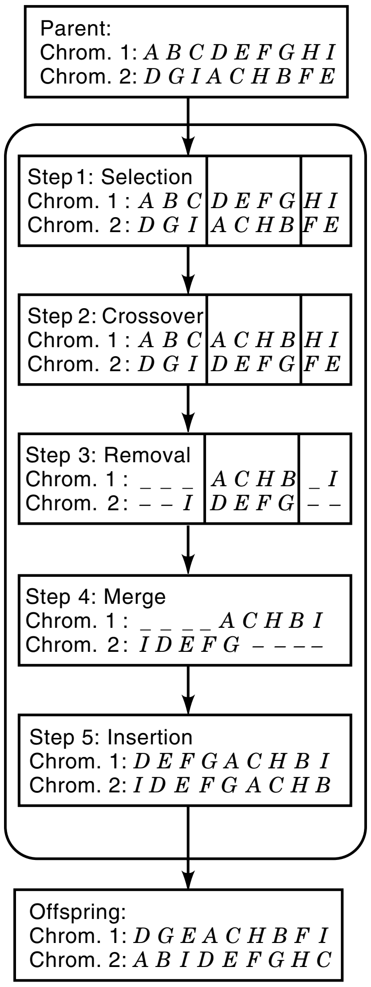

Parent:
Chrom. 1: A B C D E F G H I
Chrom. 2: D G I A C H B F E
Starting point
A B C D E F G H I
D G I A C H B F E
String 1: _ _ C _ E _ _ _ I
String 2: _ _ I _ C _ _ _ E
New 1: D G C A E H B F I
New 2: A B I D C F G H E
Offspring:
Chrom. 1: D G C A E H B F I
Chrom. 2: A B I D C F G H E

Figure 5. CX operation.



Parent:
Chrom. 1: A B C D E F G H I
Chrom. 2: D G I A C H B F E
Step 1: Selection
Chrom. 1 : A B C | D E F G | H I
Chrom. 2: D G I | A C H B | F E
Step 2: Crossover
Chrom. 1 : A B C | A C H B | H I
Chrom. 2: D G I | D E F G | F E
Step 3: Removal
Chrom. 1 : _ _ _ | A C H B | _ I
Chrom. 2: – – I | D E F G | – –
Step 4: Merge
Chrom. 1 : _ _ _ _ A C H B I
Chrom. 2: I D E F G – – – –
Step 5: Insertion
Chrom. 1: D E F G A C H B I
Chrom. 2: I D E F G A C H B
Offspring:
Chrom. 1: D G E A C H B F I
Chrom. 2: A B I D E F G H C

Figure 6. OX operation.

2. Select the city with the largest number of connections, or randomly select one in the case of a tie.
3. Select the city with the fewest connections first, to prevent its isolation.
4. This processing continues until the tour is constructed.

A time step table for the new offspring generation is shown in Table 1. Initially, *B*, *C*, *G*, and *H* contain four links. Assume that *B* is randomly selected as starting city. Then *B* will be eliminated from all the edge maps. At step 1, since *A*, *E*, and *F* contain only two links, assuming *A* is randomly selected as the second city to visit, then *A* is eliminated from all edge maps. At step 2, *C*, *E*, *F*, *I* contain two edges, assuming *F* is randomly selected, then *F* is eliminated from all edge maps. At step 3, *E* contains one link. Then *E* is selected and eliminated from all edge maps. At step 4, since *C*, *D*, *G*, *I* contain two edges, the randomly selected city is *D*; thus *D* is eliminated from all edge maps. At step 5, since *C* contains one edge, *C* is selected and eliminated from all edge maps. At step 6, *G*, *H*, and *I* contain two edges, and the randomly selected city is *I*; then *I* is eliminated from all edge maps. At step 7, since *G* and *H* contain only one edge each, the randomly selected city is *G*, which is eliminated from all edge maps. *H* becomes the last city to visit, and the tour goes back to city *B*. The tour construction becomes *B*, *A*, *F*, *E*, *D*, *C*, *I*, *G*, *H*. Comparing with the parent strings (*A B C D E F G H I* and *D G I A C H B F E*), we see that the entire edges of the offspring are taken from both parents except *A F* and *C I*. For *A F* and *C I*, two new edges are introduced in the tour. This is called *edge failure.* For a large TSP, the edge failures as reported by Whitley amounts to less than 2%, which is similar to a conventional mutation rate. Based on the experiments, Whitley proposed that rather than optimizing for the positions in order, the algorithm should probably allocate more reproductive trials to high-performance edges and find the critical edge combination.

Homaifar Matrix Crossover. Homaifar et al. (18) proposed a different TSP representation method. Instead of a string representation, a binary matrix is used to represent edges directly. The applied crossover is a conventional one. For this representation, the tour and procedure are now presented, using a nine-city problem as example:

1. Code strings into binary matrices for the parents. String 1 (*A B C D E F G H I*) and string 2 (*D G I A C H B F E*) are thus coded into 2-D matrices. Figure 7 shows the chromosomes.
2. Assuming the crossover sites are indicated by the arrows for string 1 and string 2, exchange the genes using the crossover operation. (This step is shown in Fig. 8.)
3. Any invalid tour is corrected by moving the duplicated 1's from the row to another row that does not have any 1. The correction is done to preserve existing edge as

Table 1. Time Step Table for the Edge Recombination

City	Initial	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
A	*B, I, C*	*I, C*							
B	*A, C, H, F*								
C	*A, B, D, H*	*A, D, H*	*D, H*	*D, H*	*D, H*	*H*			
D	*C, E, G*	*C, E, G*	*C, E, G*	*C, E, G*	*C, G*				
E	*D, F*	*D, F*	*D, F*	*D*					
F	*B, E, G*	*E, G*	*E, G*						
G	*D, F, H, I*	*D, F, H, I*	*D, F, H, I*	*D, H, I*	*H, I*	*H, I*	*H, I*	*H*	*H*
H	*B, C, G, I*	*C, G, I*	*C, G, I*	*C, G, I*	*C, G, I*	*C, G, I*	*G, I*	*G*	
I	*A, G, H*	*A, G, H*	*G, H*	*G, H*	*G, H*	*G, H*	*G, H*		

String 1

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	1	0	0	0	0	0	0	0
B	0	0	1	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0	0
D	0	0	0	0	1	0	0	0	0
E	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	1	0
H	0	0	0	0	0	0	0	0	1
I	1	0	0	0	0	0	0	0	0

Coding parent 1 into matrix 1

(**a**)

String 2

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	0	1	0	0	0	0	0	0
B	0	0	0	0	0	1	0	0	0
C	0	0	0	0	0	0	0	1	0
D	0	0	0	0	0	0	1	0	0
E	0	0	0	1	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0
G	0	0	0	0	0	0	0	0	1
H	0	1	0	0	0	0	0	0	0
I	1	0	0	0	0	0	0	0	0

Coding parent 2 into matrix 2

(**b**)

Figure 7. Coding of parents 1 and 2 into matrices.

String 1b

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	1	0	0	0	0	0	0	0
B	0	0	0	0	0	1	0	0	0
C	0	0	0	0	0	0	1	0	0
D	0	0	1	0	0	0	0	0	0
E	0	0	0	1	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0
G	0	0	0	0	0	0	0	1	0
H	0	0	0	0	0	0	0	0	1
I	1	0	0	0	0	0	0	0	0

String 2b

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	0	1	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	0	0	1	0
D	0	0	0	0	1	0	1	0	0
E	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1
H	0	1	0	0	0	0	0	0	0
I	1	0	0	0	0	0	0	0	0

Figure 9. First-level correcting matrices 1 (from string 1a to string 1b) and 2 (from string 2a to string 2b).

much as possible. The resulting matrices are shown in Fig. 9. The offspring from this procedure are as follows:

String 1b: *A B F E D C G H I*

String 2b: *A C D E F G I* and *B H*

4. Since the binary matrix representation may produce an illegal tour, additional correction in the binary matrix may be needed. The correction procedure is performed with the objective of preserving as many existing edges as possible. For string 2b (*A C D E F G I* and *B H*), the correction can be made by comparing the parent tour edges and modifying the string to *A B H C D E F G I*. Figure 10 shows the changes to the matrix for string 2b.

Homaifar et al. (18) suggested that the specifics of a genetic algorithm's implementation, including its representation, play an important role in its ability to satisfactorily solve the TSP or any other such problem.

Fogel Evolutionary Programming. As described earlier, evolutionary programming is a subset of evolutionary computation. Evolutionary programming emphasizes the level of the species rather than that of the chromosome operation. Genetic algorithms, in general, may be viewed as bottom-up procedures that combine building blocks of code to arrive at superior solutions, whereas evolutionary programming is a top-down procedure that tries to optimize all parameters of a cohesive interactive code simultaneously.

Fogel (19) has proposed an inversion operation for the TSP. The procedure uses *simulated annealing*. From observing natural systems, Fogel suggests that the behavior difference across generations decreases over time as species become better predictors of their environment. The approach is described below:

1. Code the TSP in an ordered list.
2. Create a population that consists of *N* parent solutions. Each parent produces a single offspring through mutation.
3. Linearly decrease the maximum inversion length from one-half of the string down to neighbor inversion at the maximum number of evaluated offspring.
4. The best *N* solutions are retained at each iteration to become parents for the next generation.

Fogel (19) suggests that it is crucial to maintain a behavioral link between parent and offspring as behavior evolves.

String 1a

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	1	0	0	0	0	0	0	0
B	0	0	1	0	0	1	0	0	0
C	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0
E	0	0	0	1	0	0	0	0	0
F	0	0	0	0	1	0	1	0	0
G	0	0	0	0	0	0	0	1	0
H	0	0	0	0	0	0	0	0	1
I	1	0	0	0	0	0	0	0	0

String 2a

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	0	1	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	0	0	1	0
D	0	0	0	0	1	0	1	0	0
E	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	1
H	0	1	0	0	0	0	0	0	0
I	1	0	0	0	0	0	0	0	0

Figure 8. Exchange of segments of matrix 1 and matrix 2 to form string 1a and string 2a.

String 2c

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	0	1	0	0	0	0	0	0
B	0	0	0	0	0	0	0	1	0
C	0	0	0	1	0	0	0	0	0
D	0	0	0	0	1	0	0	0	0
E	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1
H	0	1	0	0	0	0	0	0	0
I	1	0	0	0	0	0	0	0	0

Figure 10. Second-level correcting matrix for string 2b.

Remarks

PMX, CX, OX, edge recombination, and MX methods use a mating procedure and a crossover operator that are complex. The evolutionary improvement between parents and offspring across generations relies on the number of parents' good chromosome segments (namely, schemata) that are inherited by the offspring. Parents' search information is merged into their offspring; thus, both parents' information on the environment appears in the offspring. In the TSP, most of the offspring tours generated by those genetic permutation operators are illegitimate. A correcting procedure is usually needed to purge the offspring of inadmissible conditions. The correcting procedure produces mutationlike effects. Most of the information from parents is scrambled. The link information between cities is not preserved when an offspring is generated.

For instance, in the MX approach there are two levels of correction. The coding method applied in binary MX and inversion needs a matrix of N^2 elements to represent one tour (chromosome), where N is the number of cities. Thus, the memory requirements are proportional to $N^2 \times$ (population-size).

The operation of MX is complex; it includes expensive correcting procedures. For the first level of correction, each row and column can have only one element set to 1. This correction procedure is based on the parent connection edges; thus, a comparison of parents' edges and possible offspring correction are required. At the second level, all the connections need to be checked to prevent a subtour condition from occurring. If a subtour exists, its removal is based on the parent tour. This type of computation grows extremely fast as the problem size increases. Edge recombination constructs a chromosome according to the linkage of the parents. However, the offspring tour closely reflects the number of links of each city in the parents' tours, and new edges are created from time to time. This approach may be considered as clustering two cities as one and randomly arranging those two-city clusters. The inheritance is kept fairly well. However, the operator is complex.

The crossover operator is the most commonly used operator. Qi and Palmieri (20,21) have shown that this operator has good characteristics for searching on the solution surface. Nevertheless, this operator gives a poor performance on the TSP type of permutation problem, as a result of the inadmissible tours that occur occasionally.

In addition to the crossover operator, inversion and mutation operators are also used in genetic algorithms. They not only produce legitimate offspring tours, but also are simple operators. They seem to have potential for TSP problems.

HOLLAND'S FUNDAMENTAL THEOREM AND THE TSP SCHEMATA

The most widely used theorem to explain the behavior of genetic algorithms is Holland's fundamental theorem, which is completely explained by Goldberg (15). This theorem is based on the concept of building blocks in a chromosome structure. A good chromosome should contain many well-organized sequences of gene structures. These building blocks are called schemata. The fundamental theorem is a mathematical tool used to interpret the formation of schemata and the accumulation of good schemata in the population of chromosomes over generations. Holland's fundamental theorem and two requirements for the design of novel genetic algorithms are presented here.

Holland's fundamental theorem facilitates the investigation of the behavior of genetic algorithms. A good solution is formed in the genetic algorithms over the generations by gradually aggregating many small well-fitted gene structures, called *schemata.* The number of schemata, $M(H)$, where H is the schemata, at a given generation t is denoted as $M(H, t)$. Holland's fundamental theorem shows that in a good genetic algorithm the number of schemata in the next generation, $M(H, t + 1)$, is larger. This is due to the genetic operations such as selection, genetic operation, and mutation. The schemata can be expressed as

$$\begin{aligned} M(H, t+1) \geq M(H, t) &\times [\text{fitness improvement} \\ &\quad (\text{selection method})] \\ &\times [\text{genetic operator survival rate}] \\ &\times [\text{mutation survival rate}] \end{aligned} \tag{5}$$

A fitness function f represents a search surface. Let H be a schema, and $f(H)$ be its fitness value. Let f_{avg} be the population-average fitness value. If the selection method is to choose those chromosomes with higher fitness value than f_{avg}, the ratio of the fitness values ensures that the schema number increases if the operator survival rates are not very low:

$$M(H, t+1) = M(H, t) \times \frac{f(H)}{f_{\text{avg}}} \times (\text{operator survival rates}) \tag{6}$$

To ensure that M increases and leads to a globally optimal solution, genetic operators should be able to search the solution surface properly. From the above, two basic requirements for designing applications based on genetic algorithms (or hybrid genetic algorithms) are obtained:

1. The selection method must have a large effect on the population.
2. The genetic operators affect the searching behavior and should provide a low disruption rate.

Qi and Palmieri (20,21) provide measurements for these requirements. For requirement 1, the selection method should show that

$$\lim_{k\to\infty} \frac{\sum f_{\text{poor}}\, g_k(x_1, \ldots, x_n)}{\sum f_{\text{well}}\, g_k(x_1, \ldots, x_n)} = 0 \tag{7}$$

where k is the generation number. Equation (7) means that as the number of generations increases, the chance of getting a poor fitness chromosome approaches zero. Thus the population should attain a high concentration of well-fitted chromosomes.

As for requirement 2, an operator with searching behavior can be evaluated as a statistically independent interaction of two chromosomes (or gene segments). Assuming chromosome A with $[x_1^a, x_2^a, \ldots, x_n^a]$ and chromosome B with $[x_1^b, x_2^b, \ldots, x_n^b]$, the joint probability density function of A and B is

$$f_{x_1, x_2, \ldots, x_n}(x_1, x_2, \ldots x_n) = f_{x_1}(x_1) f_{x_2}(x_2) \cdots f_{x_n}(x_n) \tag{8}$$

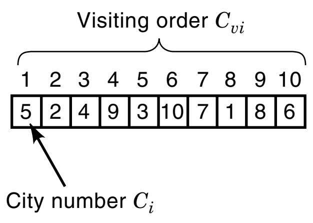


Figure 11. Ten-city TSP chromosome.

Under this condition, two chromosomes will expand their search range when a genetic operator is used. Those two chromosomes will gradually merge. Thus, their covariance will approach zero.

The disruption rate of the genetic operators is studied below along with the proposed genetic algorithms, since these operators affect the performance of the algorithms.

HYBRID NEWTON–RAPHSON GENETIC ALGORITHM

In this section an *inversion with embedded Newton–Raphson search* (IENS) is introduced. This algorithm is used as an example of genetic algorithms for the TSP. IENS uses an inversion genetic operator incorporating the Newton–Raphson method. By detecting the neighborhood tendency, the Newton–Raphson method is used to update the system variables. The algorithm combines the behavior of Newton–Raphson method, genetic algorithm neighborhood inversion, and standard inversion operators to perform a global and local search in the solution space. The fittest chromosome will become the parent of the next generation so as to seek the minimum by a Newton–Raphson update. Genetic algorithms provide powerful multiple-point search within a solution space. Through competition and reproduction, the genetic algorithm enables the Newton–Raphson operation to move around the solution surface.

Using a genetic algorithm (GA) for the solution of a problem often requires encoding the problem into a chromosome structure. An appropriate genetic-like manipulation of the chromosomes leads toward a solution of the problem. It is therefore important to choose a chromosome code that facilitates genetic-like manipulation. In this approach, a city is coded as a gene and the gene position in the chromosome is interpreted as the order of travel (adjacency representation) as introduced by Michalewicz (22).

Figure 11 shows an example of the chromosome coding for a 10-city TSP chromosome. In this example the first visited city is 5, which is followed by 2, 4, 9, 3, 10, 7, 1, 8, 6, and then 5 again. The gene pool needs be carefully selected. The gene represents the system's feature, character, or detector. The gene pool must contain sufficient information about the system. Since the TSP restriction is to visit each city exactly once and each gene represents a city, the size of the gene pool is the same as the city number. Termination is usually ensured by a ceiling on the number of generations.

For the TSP, given the chromosome representation, city ordering is the major concern. This in turn makes this problem a permutation one. The schemata are formed according to the relation between genes (cities) rather than the position within the chromosome. A permutation search genetic operator can be implemented by using an inversion. Whitely et al. (17) have used an inversion operator with reasonably good results. An inversion operator used on a 50-city TSP has outperformed a cross-and-correct operator. A drawback of using the inversion operator is that it takes information from only one parent. Thus, this approach does not provide alternative information that is often found in recombination. The search power that results from using recombination is not exploited. In order to compensate for this drawback, a numerical method has been embedded in the genetic search.

The IENS consists of five major components: the fitness function, GA operators, the operation mode, selection, and reproduction. The *fitness function,* used in this approach, is the distance of the tour. The chromosome structure provides information about the distance. A well-fitted chromosome should have a small distance (compared with its predecessors). In this approach there are three *GA operators:* inversion, neighborhood inversion, and mutation. The inversion operator randomly selects two nonneighbor genes (cities) in the chromosome and inverts the gene string. Neighborhood inversion selects two consecutive genes in the chromosome to perform a pairwise inversion. In this approach, mutation is done by randomly selecting two genes from the gene pool; these two genes swap their positions in the chromosome. The IENS *operation mode* can be summarized as having two operations: inversion search and neighbor inversion search. It is shown in Fig. 12.

The same search operation is used as long as it produces a better solution. After a threshold number of generations with no better solution, the other inversion operator is used on the best result in the history. The threshold can be determined by means of experimental runs. In the experiments and simulations the number of generations required was found to be less than 40; thus, the threshold was set to 40.

These two inversion search operations have a similar structure. The operations have five basic steps which are described below. These steps, also shown in Fig. 13, are:

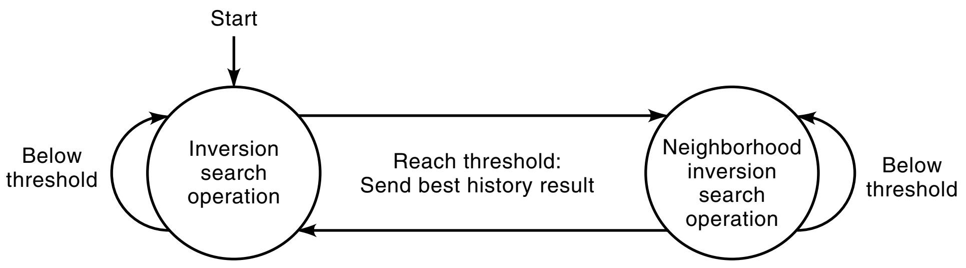

Start
Below threshold
Inversion search operation
Reach threshold: Send best history result
Neighborhood inversion search operation
Below threshold

Figure 12. IENS operation mode.

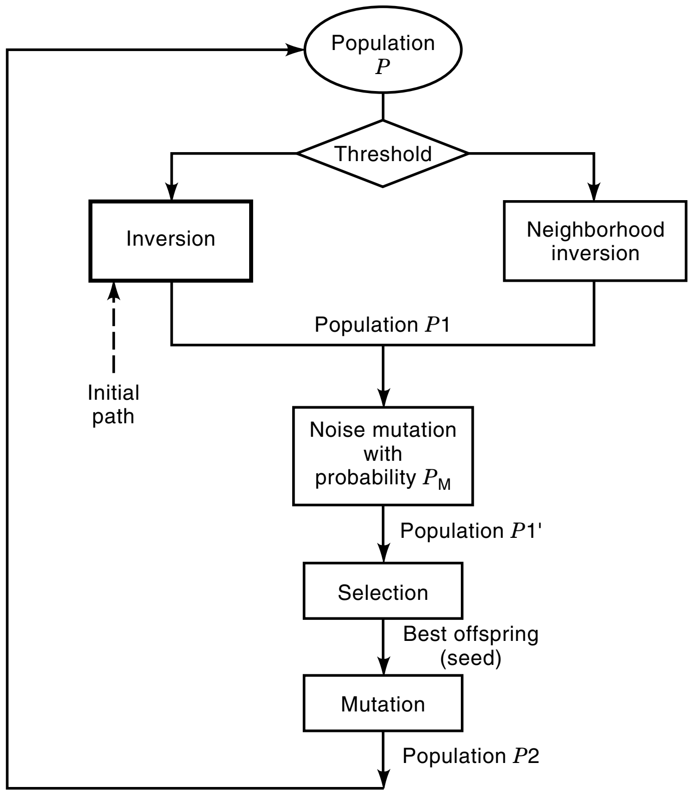


Figure 13. Inversion or neighborhood inversion operation steps.

1. Perform an inversion operation on the population P. This operation generates an offspring population P1.
2. Perform a noise mutation on offspring population P1, randomly with probability P_M. In general P_M is small, e.g., $P_M = 0.01$.
3. Evaluate P1′ and select the best offspring.
4. Generate population P2 using mutation as the reproduction operator with the best offspring as seed.
5. Replace P with P2.

The neighborhood inversion search operation is accomplished in a similar fashion to inversion search, except that the first step is replaced by the neighborhood inversion operation as shown in Fig. 13. Each parent can generate only one offspring. In both search operations an approach has been embedded to find the solution.

This approach mimics the Newton–Raphson method (23), numerical method used to solve the problem of making n functional relations equal to zero: $F_i(x_1, x_2, \ldots, x_m) = 0$, where $i = 1, 2, \ldots, n$. If X denotes the entire vector of values of x_i, then, in the neighborhood of X, each of the functions F_i can be expanded into a Taylor series as follows:

$$F_i(X + \delta X) = F_i(X) + \sum_{j=1}^{m} \frac{\partial F_i}{\partial x_j}\delta x_j + O(\delta X^2) \qquad (9)$$

By neglecting terms of order δX^2 and higher, a set of linear equations can be obtained. A change δX moves each function closer to zero when the system converges. That is,

$$\sum_{j=1}^{m} \alpha_{ij}\,\delta x_j = \beta_i, \qquad \text{where} \quad \alpha_{ij} = \frac{\partial F_i}{\partial x_j} \text{ and } \beta_i = -F_i \qquad (10)$$

If δX has a valid (finite) value, the correction becomes $x_i^{\text{new}} = x_i^{\text{old}} + \delta x_i$.

In the proposed genetic algorithm, each city is coded as a gene. Thus, F becomes a chromosome. The distance of a tour is given by $F_i(x_1, x_2, \ldots, x_m)$, where each x represents a city. If we denote the minimum distance by $D_{\min}$, then $F_i(x_1, x_2, \ldots, x_m) = D_{\min}$ represents the solution. The coefficient α_{ij} in the genetic algorithm represents a neighborhood inversion and is given by a derivative term. The neighborhood inversion acts as a local hill-climbing operator. For a given generation, i and j represent the ith chromosome and the jth gene. δx is a continuous function and provides information for the next solution. Since δx has discrete values in the genetic algorithm environment, a steepest-descent approach for δx can be used. Thus, x_i^{new} is equal to the maximum improvement in the β_i, that is, F_i. Since a steepest-descent approach is applied, the selection method is simplified to selecting only the best chromosome. To achieve reproduction, mutation is applied to generate the entire population from the best offspring. Using mutation prevents the population from settling far away from the current point. The population size has been set to be equal to the city problem size, i.e., $n = m$; the Newton–Raphson method needs the number of independent equations to be at least equal to the number of variables to have a solvable system.

To use the Newton–Raphson method, two major issues need be addressed: (1) the starting point must be near the solution and (2) the operation can easily be trapped in a local minimum. To avoid these problems, standard inversion is applied in the search process. Standard inversion provides large steps that can help not only to reach a near-optimal point but also to jump out of a trap (local minimum).

Each operation needs at least several generations to reach the solution or a stable state; thus, a generation threshold for each operation and generation counter must be set. A record of the best history is always kept. If the search result is better than the record, the record is updated and the generation counter is reset. If the search result is not better than the previous result, the generation counter is increased by one. When the generation counter reaches the generation threshold, the record is transferred to the other operation as the starting seed. When the other operation receives the seed, mutation is applied to generate the new population. Since there is no information about where the optimal solution resides, the initial chromosome is randomly generated.

Dynamics

Sirag and Weisser (24) have shown that when the inversion operates on a string chromosome, the behavior of this operator can be obtained by evaluating the survival rate of the O schemata. The O-schema number, as explained in Goldberg (15), provides information about the goodness of the solution. This approach is used to show the dynamics of IENS. It is assumed that the chromosome length is L and two distinct points A and B are selected to represent the ends of the inversion.

Standard Inversion. For standard inversion, A could be either larger or smaller than B. The expected survival rate E_A for a given A is expressed as

$$E_A = P(A > B|A)E_{A>B} + P(A < B|A)E_{A<B} \tag{11}$$

where $E_{A>B}$ and $E_{A<B}$ are the expected O-schema survival rates under the condition $A > B$ and $A < B$, respectively. For simplicity, the probabilities $P(A > B|A)$ and $P(A < B|A)$ will be abbreviated to $P(A > B)$ and $P(A < B)$ respectively. These probabilities are

$$P(A > B) = \frac{A-1}{L-1} \quad \text{and} \quad P(A < B) = \frac{L-A}{L-1} \tag{12}$$

The total number of O schemata can be considered as the total number of subsets; thus, for chromosomes of size L, the total number of O schemata is 2^L. Then $E_{A>B}$ can be calculated by using the total number of O schemata minus the number of O schemata with genes in the inversion range. The result is multiplied by the probability for a gene to be selected as $B[P = 1/(A - 1)]$. Thus,

$$E_{A>B} = \frac{1}{A-1}[2^L - (2^{1+A-B} - 1)] \tag{13}$$

$E_{A<B}$ can be calculated in a similar fashion:

$$E_{A<B} = \frac{1}{L-A}[2^L - (2^{1+B-A} - 1)] \tag{14}$$

By replacing B with x, we obtain the expected value of $E_{A>B}$ as

$$\begin{aligned} E_{A>B} &= \sum_{x=1}^{A}\left(\frac{1}{A-1}[2^L - (2^{1+A-x} - 1)]\right) \\ &= \frac{1}{A-1}\left(2^L(A-1) + (A-1) - \sum_{x=2}^{A} 2^x\right) \end{aligned}$$

Likewise, the expected value of $E_{A<B}$ is

$$\begin{aligned} E_{A<B} &= \sum_{x=1}^{1+L-A}\left(\frac{1}{L-A}[2^L - (2^{1+x-A} - 1)]\right) \\ &= \frac{1}{L-A}\left(2^L(L-A) + (L-A) - \sum_{x=2}^{1+L-A} 2^x\right) \end{aligned}$$

The products of the probabilities and expected values are expressed as

$$\begin{aligned} P(A > B)E_{A>B} &= \frac{A-1}{L-1} \\ &\quad \times \frac{1}{A-1}\left(2^L(A-1) + (A-1) - \sum_{x=2}^{A} 2^x\right) \\ P(A < B)E_{A<B} &= \frac{L-A}{L-1} \\ &\quad \times \frac{1}{L-A}\left(2^L(L-A) + (L-A) - \sum_{x=2}^{1+L-A} 2^x\right) \end{aligned}$$

Since $E_A = P(A > B)\,E_{A>B} + P(A < B)\,E_{A<B}$,, then

$$E = \sum_{A=1}^{L} E_A \tag{15}$$

Thus

$$\begin{aligned} E &= \left[\sum_{A=1}^{L} \frac{L}{L-1}\left(2^L(A-1) + (A-1) - \sum_{x=2}^{A} 2^x\right)\right. \\ &\quad \left. + \left(2^L(L-A) + (L-A) - \sum_{x=2}^{1+L-A} 2^x\right)\right] \\ &= \sum_{A=1}^{L}\left(2^L + 1 - \frac{1}{L-1}(2^{A+1} - 4 + 2^{2+L-A} - 4)\right) \\ &= L(2^L + 1)\frac{1}{L-1}\left(\sum_{A=1}^{L} 2^{A+1} + \sum_{A=1}^{L} 2^{2+L-A} - 8L\right) \\ &= L(2^L + 1)\frac{1}{L-1}(8 \times 2^L - 8 - 8L) \end{aligned}$$

Thus,

$$E = L \times (2^L + 1) - \frac{1}{L-1}(8 \times 2^L - 8 - 8L)$$

and $E_{\text{avg}} = E/L$, so that

$$E_{\text{avg}} = (2^L + 1) + \frac{8}{L-1} + \frac{8}{L(L-1)} - \frac{8 \times 2^L}{L(L-1)}$$

Thus, the survival rate is

$$\text{SR}_{\text{inv}} = \frac{E_{\text{avg}}}{2^L} = 1 + \frac{1}{2^L} + \frac{8}{2^L(L-1)} + \frac{8}{2^L L(L-1)} - \frac{8}{L(L-1)}$$

For a large chromosome, L becomes large. Thus, the survival rate can be simplified to

$$\text{SR}_{\text{inv}} = 1 - \frac{8}{L(L-1)} \tag{16}$$

Neighborhood Inversion. Neighborhood inversion can be considered as a special case of standard inversion. Since only one component will be selected, the selection probabilities become

$$P(A > B) = \tfrac{1}{2} \quad \text{and} \quad P(A < B) = \tfrac{1}{2}$$

The equations (13) and (14) become

$$\begin{aligned} E_{A>B} &= 2^L - (2^2 - 1) \\ E_{A<B} &= 2^L - (2^2 - 1) \end{aligned}$$

Using Eq. (11) results in $E_A = 2^L - (2^2 - 1)$. Thus, the survival rate becomes

$$\text{SR}_{\text{n inv}} = 1 - \frac{3}{2^L} \tag{17}$$

Mutation. Mutation behaves similarly to standard inversion from the expected-value point of view. Thus, the expected value is given by Eq. (11), and the selection probabilities are given by Eq. (12). $E_{A>B}$ can be calculated by using the total number of O schemata minus the number of O schemata that use genes in the inversion range. The result is multiplied by the probability $P = 1/(A - 1)$ for a gene to be selected as B. Thus, $E_{A>B}$ is

$$E_{A>B} = \frac{1}{A-1}[2^L - (2^2 - 1)]$$

In a similar fashion $E_{A<B}$ is calculated using $P = 1/(L - A)$:

$$E_{A<B} = \frac{1}{L-A}[2^L - (2^2 - 1)]$$

The PE products become

$$P(A > B)E_{A>B} = \frac{A-1}{L-1} \times \frac{1}{A-1}[2^L \times (A-1) - 3 \times (A-1)]$$

and

$$P(A < B)E_{A<B} = \frac{L-A}{L-1} \times \frac{1}{L-A}[2^L \times (L-A) - 3 \times (L-A)]$$

Thus, using Eq. (17), E becomes

$$\begin{aligned} E &= \sum_{A=1}^{L}\left(\frac{1}{L-1}[2^L(A-1) - 3(A-1)] \right. \\ &\quad \left. + \frac{1}{L-1}[2^L(L-A) - 3(L-A)]\right) \\ &= \sum_{A=1}^{L}(2^L - 6) \\ &= L \times 2^L - 6L \end{aligned}$$

The average E is

$$E_{\text{avg}} = \frac{E}{L} = 2^L - 6$$

and the survival rate becomes

$$\text{SR}_{\text{mut}} = 1 - \frac{6}{2^L} \quad (18)$$

Analysis Using Holland's Fundamental Theorem. Holland's fundamental theorem is used to investigate the behavior of IENS. The selection method is to choose the fittest chromosome as seed; therefore, the next-generation O-schema number $M(H, t + 1)$ is

$$M(H, t+1) = M(H, t)\frac{f(H)}{f_{\text{avg}}} \quad (19)$$

where H is the O schema, $f(H)$ is the fitness value, and f_{avg} is the population-average fitness value. The ratio of the fitness values ensures that the O-schema number increases. The reproduction is done by means of the mutation operator; thus Eq. (19) becomes

$$M(H, t+1) = M(H, t)\frac{f(H)}{f_{\text{avg}}}\left(1 - \frac{6}{2^L}\right) \quad (20)$$

For the inversion search operation, $M(H, t + 1)$ satisfies the following inequality:

$$\begin{aligned} M(H, t+1) \geq M(H, t)&\frac{f(H)}{f_{\text{avg}}}\left(1 - \frac{6}{2^L}\right) \\ &\times \left(1 + \frac{1}{2^L} + \frac{8}{2^L(L-1)} + \frac{8}{2^L L(L-1)} \right. \\ &\left. - \frac{8}{L(L-1)} - \frac{6P}{2^L}\right) \end{aligned}$$

When L is large, this can be simplified to

$$M(H, t+1) \geq M(H, t)\frac{f(H)}{f_{\text{avg}}}\left(1 - \frac{8}{L(L-1)}\right) \quad (21)$$

For the neighborhood inversion search operation, $M(H, t + 1)$ satisfies the following inequality:

$$M(H, t+1) \geq M(H, t)\frac{f(H)}{f_{\text{avg}}}\left(1 - \frac{6}{2^L}\right)\left(1 - \frac{3}{2^L} - \frac{6P}{2^L}\right)$$

For large L, this is simplified to

$$M(H, t+1) \geq M(H, t)\frac{f(H)}{f_{\text{avg}}} \quad (22)$$

To ensure that M increases as the equilibrium point gets closer, a comparison between $f(H, t)$ at time t and $f(H, t + 1)$ at time $t + 1$ is necessary. The best history of each operation is transferred to the other as seed; this in turn provides a monotonic change of M.

Effect of Operators on Distance

Changes in M depend largely on the selection, as shown earlier. The inversion and mutation operators provide a slow and steady improvement. The effect of these operators on the traveled distance is evaluated in this subsection.

Inversion. For the inversion operator two points must be selected. The probability of randomly selecting two points from a string of length m is $\binom{m}{2}^{-1}$. A uniform distribution is used, and the chromosome is considered to be a ring string rather than a line string. Assuming that A and B are the two inversion points, then $|A - B| + 1$ is the number of genes to be inverted in the string. Having $A = i$ and $B = j$, the original distance and the new distance after inversion are

$$\begin{aligned} \text{Dist}_{\text{orig}} &= d_{1,2} + d_{2,3} + \cdots + d_{i-1,i} + d_{i,i+1} + \cdots + d_{j-1,j} \\ &\quad + d_{j,j+1} + \cdots + d_{m-1,m} + d_{m,1} \\ \text{Dist}_{\text{inv}} &= d_{1,2} + d_{2,3} + \cdots + d_{i-1,j} + d_{j,j-1} + \cdots + d_{i+1,i} \\ &\quad + d_{i,j+1} + \cdots + d_{m-1,m} + d_{m,1} \end{aligned}$$

The new distance Dist_{inv} has been modified in $|A - B| + 1$ genes. However, the overall distance change is equal to $|d_{i-1,i} + d_{j,j+1} - d_{i-1,j} - d_{i,j+1}|$. This is because in the considered TSP

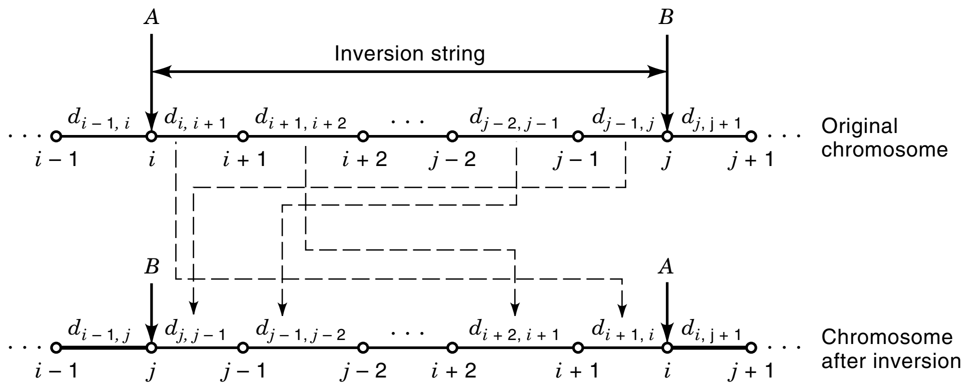


Figure 14. Changes in the chromosome edges after using inversion.

the distance calculation is direction-insensitive, i.e., $d_{i,i+1} = d_{i+1,i}$. It should be pointed out that there are other TSPs where distance is direction-sensitive ($d_{i,i+1} \neq d_{i+1,i}$).

Figure 14 shows the changes in the chromosome edges when the inversion operation is applied. The new links in this chromosome are drawn with thicker lines. The new distance is rewritten as

$$\begin{aligned}\text{Dist}_{\text{inv}} &= d_{1,2} + d_{2,3} + \cdots + d_{i-1,j} + d_{i,i+1} + \cdots + d_{j-1,j}\\ &\quad + d_{j,j+1} + \cdots + d_{m-1,m} + d_{m,1}\end{aligned}$$

Thus, Dist_{inv} can be expressed in terms of the original distance as follows:

$$\text{Dist}_{\text{inv}} = \text{Dist}_{\text{orig}} - d_{i-1,i} - d_{j,j+1} + d_{i-1,j} + d_{i,j+1}$$

on replacing i by A and$j$ by B, the distance expression becomes

$$\text{Dist}_{\text{inv}} = \text{Dist}_{\text{orig}} - d_{A-1,A} - d_{B,B+1} + d_{A-1,B} + d_{A,B+1}$$

In order to obtain the upper and lower bounds, distances d_{orig} and d_{inv} need be introduced:

$$\begin{aligned}d_{\text{orig}} &= \max(d_{A-1,A}, d_{B,B+1})\\ d_{\text{inv}} &= \max(d_{A-1,B}, d_{A,B+1})\end{aligned}$$

Thus Dist_{inv} becomes

$$\text{Dist}_{\text{inv}} = \text{Dist}_{\text{orig}} - 2d_{\text{orig}} + 2d_{\text{inv}}$$

Thus the upper and lower bounds of the distance after inversion become

$$\text{Dist}_{\text{orig}} - 2d_{\text{orig}} \leq \text{Dist}_{\text{inv}} \leq \text{Dist}_{\text{orig}} + 2d_{\text{inv}}$$

Considering a 2-D Euler surface, the maximum distance between two points in a unit square is

$$d_{\max} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = \sqrt{2} \qquad (23)$$

Therefore, the bounds for Dist_{inv} are $\text{Dist}_{\text{orig}} - 2\sqrt{2} \leq \text{Dist}_{\text{inv}} \leq \text{Dist}_{\text{orig}} + \sqrt{2}$.

For an r-square surface, the following changes are introduced: x_1 to rx_1, x_2 to rx_2, y_1 to ry_1, and y_2 to ry_2. Consequently, the distance bounds become

$$\text{Dist}_{\text{orig}} - 2r\sqrt{2} \leq \text{Dist}_{\text{inv}} \leq \text{Dist}_{\text{orig}} + 2r\sqrt{2} \qquad (24)$$

Mutation. For mutation distance analysis a similar approach and assumptions to those of the inversion analysis can be used. Having $A = i$ and $B = j$, the original distance and the new distance after mutation are

$$\begin{aligned}\text{Dist}_{\text{orig}} &= d_{1,2} + d_{2,3} + \cdots + d_{i-1,i} + d_{i,i+1} + \cdots + d_{j-1,j}\\ &\quad + d_{j,j+1} + \cdots + d_{m-1,m} + d_{m,1}\\ \text{Dist}_{\text{mut}} &= d_{1,2} + d_{2,3} + \cdots + d_{i-1,j} + d_{j,i+1} + \cdots + d_{j-1,i}\\ &\quad + d_{i,j+1} + \cdots + d_{m-1,m} + d_{m,1}\end{aligned}$$

The new distance has been modified only on two genes. However, the overall distance change is equal to $|d_{i-1,i} + d_{i,i+1} +$

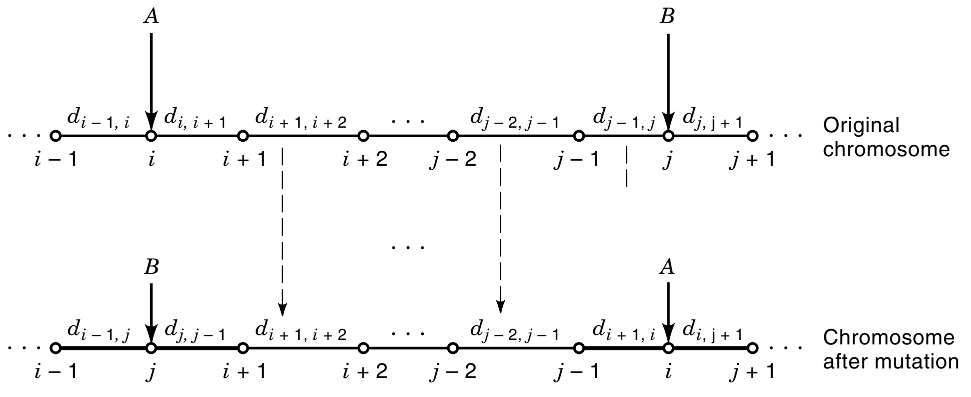


Figure 15. Changes in the chromosome edges after using mutation.

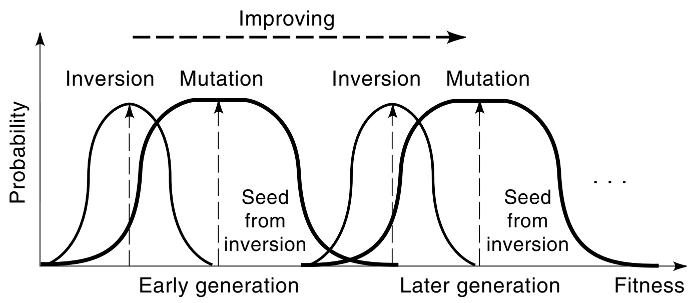


Figure 16. Monotonic behavior of the proposed IENS genetic algorithm.

$d_{j-1,j} + d_{j,j+1} - d_{i-1,j} - d_{j,i+1} - d_{j-1,i} - d_{i,j+1}|$. This is because a mutation operation changes four edges.

Figure 15 shows changes in the chromosome edges when the mutation is applied. This figure highlights the affected links. On replacing i by A and j by B, the distance expression becomes

$$\begin{aligned}\text{Dist}_{\text{mut}} = \text{Dist}_{\text{orig}} &- d_{A-1,A} - d_{A,A+1} - d_{B-1,B} - d_{B,B+1} \\ &+ d_{A-1,B} + d_{B,A+1} + d_{B-1,A} + d_{A,B+1}\end{aligned}$$

In order to obtain the upper and lower bounds, two distances (d_{orig} and d_{mut}) are introduced:

$$d_{\text{orig}} = \max(d_{A-1,A}, d_{A,A+1}, d_{B-1,B}, d_{B,B+1})$$
$$d_{\text{mut}} = \max(d_{A-1,B}, d_{B,A+1}, d_{B-1,A}, d_{A,B+1})$$

Thus Dist_{mut} becomes

$$\text{Dist}_{\text{mut}} = \text{Dist}_{\text{orig}} - 4d_{\text{orig}} + 4d_{\text{mut}}$$

Thus the upper and lower bounds of the distance after mutation become

$$\text{Dist}_{\text{orig}} - 4d_{\text{orig}} \leq \text{Dist}_{\text{mut}} \leq \text{Dist}_{\text{orig}} + 4d_{\text{mut}}$$

Considering a 2-D Euler unit square and the distance $d_{\max}$ in Eq. (23), from Eq. (24) the bounds for Dist_{mut} are $\text{Dist}_{\text{orig}} - 4\sqrt{2} \leq \text{Dist}_{\text{mut}} \leq \text{Dist}_{\text{orig}} + 4\sqrt{2}$. For an r-square surface, this becomes

$$\text{Dist}_{\text{orig}} - 4r\sqrt{2} \leq \text{Dist}_{\text{mut}} \leq \text{Dist}_{\text{orig}} + 4r\sqrt{2} \qquad (25)$$

Behavior

It is interesting to observe how the IENS approach reaches the solution. Using Eqs. (20) to (25), it is possible to illustrate the behavior of IENS as function of fitness.

The monotonic behavior of IENS is shown in Fig. 16. It can be observed that the inversion and mutation operators drive the chromosome population towards better fitness values.

Since the best chromosome after inversion is preserved as the seed for reproduction (using mutation), the distance is driven toward better fitness. Each of the reproduced chromosome can obtain a fitness value that is as much as twice its parent's fitness value. The reproduced chromosomes are then used by the inversion operator.

The inversion and neighborhood inversion searches have similar effects on the distance. Thus, these two operators have comparable distributions. The best-history chromosome is kept as the starting seed for each search operator. From the fundamental theory analysis for IENS and the distance analysis, both the ordering survival rate and number of O schemata can be estimated as well as how much the distance is improved. IENS preserves the order and at the same time explores the search solution space in a monotonic fashion.

Performance

The IENS approach has been applied to a number of TSP benchmarks. These benchmarks include 10-city problems (25), 30-city problems (16), 50- and 75-city problems (26), and 105- and 318-city problems (11).

For the 10-city TSP, a set of five benchmarks, fully described in Lin et al. (10,25), have been used. The hybrid genetic algorithm has been used to find solutions for these benchmarks. A summary of the results is shown in Table 2. The optimal solutions are included. The ceiling number of generations is set at 200. The threshold number of generations is 5. For each benchmark 100 runs were performed. From Table 2, it can be observed that:

- The IENS algorithm obtains the optimal solution for all the 10-city TSPs in every run. The distance is that of the best solution, i.e., at least one chromosome per run represents the optimal solution.
- The number of generations to find the optimal solutions in the best case is extremely small. It ranges from 3 to 7.
- The average number of generations to find the optimal solution is fairly small. With exception of benchmark 10.4, it is below 30.

Other benchmarks for 30- to 318-city TSPs were used to evaluate the proposed IENS approach. The results of the simulations are shown in Table 3. For comparison, the results are reported using both the integer and real-number solutions. The integer solutions are obtained by adding the roundoff distance between any two cities in the traveled path. The percentage divergence rate gives a measurement of the difference

Table 2. Simulation Results for 10-City TSP Benchmarks

Benchmark:	10.1	10.2	10.3	10.4	10.5
Optimal distance:	2.986918	3.52247	2.82804	2.88463	2.9262
IENS best solution:	2.986918	3.52247	2.82804	2.88463	2.9262
IENS best-solution generation:	5	3	6	7	4
IENS average distance:	2.986918	3.52247	2.82804	2.88463	2.9262
IENS average generations:	21.58	20.38	23.94	69.47	26.04

Table 3. Results for 30- to 318-City TSP Problems

Benchmark:	30	50	75	105	318
Best known solution:	420 (integer)	425 (integer)	535 (integer)	14382.9	41345 (integer)
IENS best solution (real):	423.740	427.855	542.309	14382.995	43105.6048
IENS best solution (int.):	420	425	535	—	43020
IENS best-soln. gen.:	262	8938	48353	25794	942052
IENS average distance:	425.75	433.821	550.849	14742.736	43710.278
IENS avg. generations:	2027.78	5899.68	50807.88	45287.16	647305.01
IENS ceiling generations:	5000	10,000	100,000	100,000	1000,000
IENS runs:	50	50	50	30	18
IENS divergence rate (%):	0.47	1.39	1.57	2.5	5.4

between the average distance and the best known solution. The percentage divergence rate is computed as follows:

$$\text{divergence rate} = \frac{\text{average distance} - \text{best known distance}}{\text{best known distance}} \times 100\% \tag{26}$$

It can be observed that:

- IENS obtains the best known solution for all the benchmarks with the exception of the 318-city benchmark.
- All the results are always extremely close to the best known solutions. Average divergence rates vary from 0.47% to 5.4% as for 30- to 318-city benchmarks.
- The average number of generations increases fast from the 30- to the 318-city benchmark. IENS requires more search time for large problems. However, due to its simplicity, IENS operates fast in each generation.

Remarks

The hybrid genetic algorithm called *inversion with embedded Newton–Raphson search* (IENS), which combines the Newton–Raphson numerical method and a genetic algorithm, has been introduced. This algorithm is an example of hybrid genetic algorithms for the TSP. The operations and their analysis can be used for other TSP genetic approaches. The IENS approach has the following characteristics:

- Simple and inexpensive operation
- Monotonic improvement of the tour
- Optimal solution for all the 10-city TSPs at every run. The distance is identical to that of the optimal solution.
- Best known solution for all the TSP benchmarks except the 318-city benchmark. It can be observed (in Table 3) that the solutions are always extremely close to the optimal solutions. The average distance increases from 0.47% to 5.4% as one goes from 30- to 318-city benchmarks.

This example shows that genetic algorithms have a potential for obtaining extremely good solutions to optimization problems such as the TSP.

BIBLIOGRAPHY

1. C. H. Papadimitriou and K. Steiflitz, *Combinatorial Optimization: Algorithms and Complexity,* Englewood Cliffs, NJ: Prentice-Hall, 1982.
2. D. J. Wilde and C. S. Beightler, *Foundations of Optimization,* Englewood Cliffs, NJ: Prentice-Hall, 1967.
3. D. A. Pierre, *Optimization Theory with Applications,* New York: Wiley, 1969.
4. A. Torn and A. Zilinxkas, *Global Optimization,* New York: Springer-Verlag, 1987.
5. E. L. Lawler et al., *The Traveling Salesman Problem,* New York: Wiley, 1985.
6. J. J. Hopfield and D. W. Tank, Neural computation of decisions in optimization problems, *Biol. Cybern.,* **52**: 141–152, 1985.
7. S. V. B. Aiyer, Solving combinatorial optimization problems using neural networks with applications in speech recognition, Ph.D. dissertation, Cambridge University, England, 1991.
8. E. Sanchez-Sinencio and C. Lau, *Artificial Neural Networks Paradigms, Applications, and Hardware Implementations,* New York: IEEE Press, 1992.
9. P. W. Pretzel, D. L. Palumbo, and M. K. Arras, Performance and fault-tolerance of neural networks for optimization, *IEEE Trans. Neural Netw.,* **4**: 600–614, 1993.
10. W. Lin, High quality tour hybrid genetic schemes for TSP optimization problems, Ph.D. dissertation, State University of New York at Binghamton, 1995.
11. S. Lin and B. W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, *Oper. Res.,* **21**: 498–516, 1976.
12. D. B. Fogel and L. J. Fogel, Evolutionary computation, *IEEE Trans. Neural Netw.,* **5**: 1, 1994.
13. D. B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Trans. Neural Netw.,* **5**: 3–14, 1994.
14. M. Srinivas and L. M. Patnaik, Genetic algorithms: A survey, *IEEE Comput.,* **27**: 17–26, 1994.
15. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, MA: Addison-Wesley, 1989.
16. I. M. Oliver, D. J. Smith, and J. R. C. Holland, A study of permutation crossover operators on the traveling salesman problem, *Proc. 2nd Int. Conf. Genet. Algorithms: Genet. Algorithms Appl.,* Cambridge, MA, 1987, pp. 224–230.
17. D. Whitley, T. Starkweather, and Q. Fuquay, Scheduling problems and traveling salesman: The genetic edge recombination operator, *Proc. 3rd Int. Conf. Genet. Algorithms: Genet. Algorithms Appl.,* Arlington, VA, 1989, pp. 133–140.
18. A. Homaifar, S. Guan, and G. E. Liepins, A new approach on the traveling salesman problem by genetic algorithms, *Genet. Algorithms Appl.: Proc. 5th Int. Conf. Genet. Algorithms,* 1993, pp. 460–466.
19. D. B. Fogel, Applying evolutionary programming to selected traveling salesman problems, *Cybern. Syst.: Int. J.,* **24**: 27–36, 1994.
20. X. Qi and F. Palmieri, Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. Part

I: Basic properties of selection and mutation, *IEEE Trans. Neural Netw.,* **5**: 102–119, 1994.

21. X. Qi and F. Palmieri, Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. Part II: Analysis of diversification role of crossover, *IEEE Trans. Neural Netw.,* **5**: 120–129, 1994.
22. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* Berlin: Springer-Verlag, 1992.
23. W. H. Press et al., *Numerical Recipes,* Cambridge, England: Cambridge University Press, 1986.
24. D. J. Sirag and P. T. Weisser, Toward a unified thermodynamic genetic operator, *Genet. Algorithms Appl.: Proc. 2nd Int. Conf. Genet. Algorithms,* 1987, pp. 116–122.
25. W. Lin et al., An evaluation of energy function for a neural network model for optimization problems, *IEEE Int. Jt. Conf. Neural Netw.,* Orlando, FL, 1994, pp. 4518–4522.
26. S. Eilon and N. Christofides, Distribution management: Mathematical modeling and practical analysis, *Oper. Res. Q.,* **20**: 309, 1969.

WEI LIN
Coopers & Lybrand LLP
JOSÉ G. DELGADO-FRIAS
DONALD C. GAUSE
State University of New York at Binghamton

TRAVELING WAVE AMPLIFIERS. See DISTRIBUTED AMPLIFIERS; TRAVELING WAVE TUBES.