# Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation

Göktürk Üçoluk

Department of Computer Engineering
Middle East Technical University
06531 Ankara, Turkey

Email: ucoluk@ceng.metu.edu.tr

**Abstract**

In this work an alternative representation for permutations which is robust under ordinary crossover and mutation, is proposed. This method is used for solving the TSP.

## 1 Introduction

A well known computational problem is the the Travelling Salesman Problem (TSP) which is known to be $NP-complete$. Here is the wording of it:

> *N points ('cities'), as well as the cost of travelling between every pair of them is given. Assume that a salesperson, starting from a given city, has to visit each city exactly once and hence make a round-trip. The aim is to find an optimal tour in which the total cost of the round-trip is minimized.*

More formally, the TSP can be formulated as a problem of graph theory: Given a graph $G$ on a set of $N$ vertices, a closed sequence of edges in $G$ (i.e. a cycle) which passes through each vertex of $G$ exactly once is called a *Hamiltonian Cycle*. Given a complete weighted graph $G$ on a set of $N$ vertices (cities) the TSP is then the problem of finding the shortest Hamiltonian Cycle through $G$. From the computational point of view this means the determination of the particular permutation of the non-repeating sequence $1, 2, \ldots, N$ where the cities are numbered consecutively from 1 to $N$ and the permutation represents the visiting order for which the weight sum is minimized.

The search space contains $N!$ permutations and since TSP is $NP-complete$ and the corresponding optimization problems are therefore NP-hard. The best known algorithms have exponential (deterministic) run time complexity.

Such combinatorial optimization problems are in the domain of Genetic Algorithms interest. In the next section, the most popular method among conventional GA solutions of TSP will be reviewed. In section **??** an alternative solution will be introduced. A comparative experimental study will be overviewed in **??** which is followed by the conclusion.

## 2 Conventional Approach

In the conventional approach a chromosome which is devised to represent a solution constitutes of $N$ (count of the cities) genes. Each gene holds a number which is a label of a city. So the $n$ th gene holds the label of the city which is visited $n$ th. In other words, the chromosome is a direct coding of a permutation of the sequence $1, 2, \ldots, N$.

The problem with this representation is obvious. Starting with a population of valid chromosomes, ordinary crossover and mutation operators cause problems. This is so because offsprings generated by means of the ordinary operators are of a great possibility no more valid chromosomes. A variety of methods and new operators that handle that sort of obscenities are introduced throughout the literature.

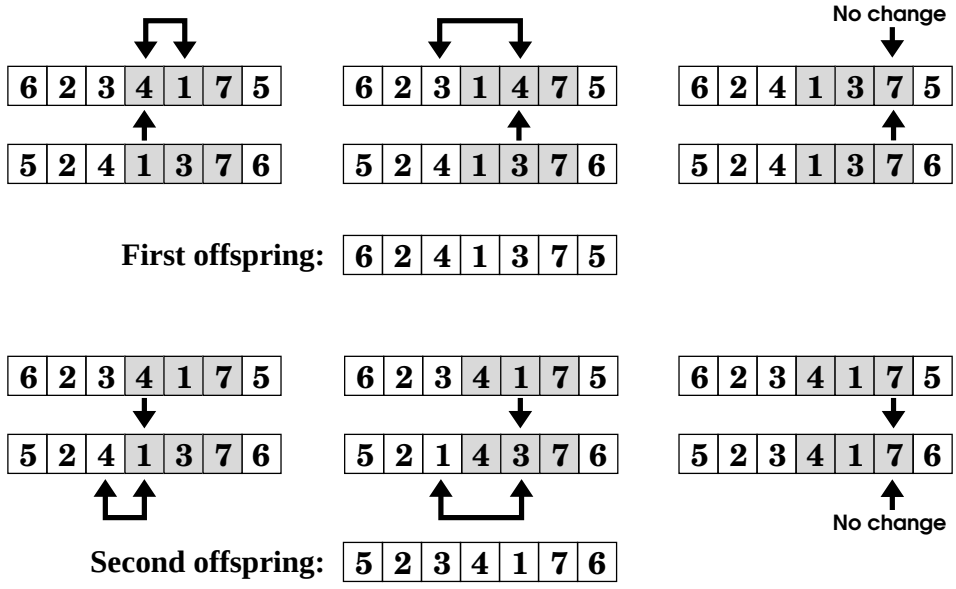
Solutions are observed to fall into one of the following three categories:

- **Disqualification:** The idea is to allow the generation of those invalid chromosomes but assign such a low fitness values that they got eliminated in the forthcoming selection process. This simple method has its disadvantage of being extensively time consuming. The genetic engine spends most of its time generating invalid chromosomes and then eliminating them.
- **Repairing:** In this approach invalid chromosomes are generated but then fed into a intermediate process where they are transformed into valid ones. Here the key idea is to do the least modification such that the merits of crossover is preserved.
- **Inventing Specialized Operators:** Instead of creating invalid chromosomes the GA operators are modified to generate only valid chromosomes.

Falling into the third category and concerning permutation-respecting crossover operators, the following operators are worth to mention:

- Partially mapped crossover (PMX). [?]
- Order crossover (OX). [?]
- Edge recombination crossover (ERX). [?]

Among these the first one, namely PMX is quoted to be the most successful. Given two parents $s$ and $t$, PMX randomly picks two crossover points - like 2-point crossover. The child is then constructed in the following way. Starting with a copy of $s$, the positions between the crossover points are, one by one, set to the values of $t$ in these positions. To keep the string a valid chromosome the cities in these positions are not just overwritten. To set position $p$ to city $c$, the city in position $p$ and city $c$ swap positions. Below you see an example of this coding and special crossover technique for two sample permutations: $6, 2, 3, 4, 1, 7, 5$ and $5, 2, 4, 1, 3, 7, 6$



The resulting child has

1. between the crossover points the same cities in the same positions as $t$, and
2. outside the crossover interval the same cities in the same positions as $s$, where this is not in conflict with (1).

This idea can very easily be generalized to $n$-point crossover. Mutation is done by exchanging gene values in pairs (in a chromosome).

This method has the following draw backs:

1. The changes in the chromosomes are not confined to the exchanged portions. Therefore the *building-blocks* mechanism of EC is damaged.
2. Mutations are not performed at single points.
3. Simple bit-string crossover and mutation implementations will not work.
4. The implementation requires tricks for efficiency purposes.

In the following section a new technique for GA to deal with permutation encoding, where the representation is crossover and mutation robust, is introduced. This means that the off-springs generated by crossover and mutation are still valid chromosomes and no special definitions for these operators are needed: the conventional bit-string crossover and mutation operators suffices.

# 3 Proposed Method

The proposed method is to describe a permutation by means of its inversions [**?**]. For a permutation $i_1, i_2, \ldots, i_N$ of the set $\{1, 2, \ldots, N\}$ we let $a_j$ denote the number of integers in the permutation which precede $j$ but are greater than $j$. So, $a_j$ is a measure of how much out of order $j$ is. The sequence of numbers $a_1, a_2, \ldots, a_N$ is called the *inversion sequence* of the permutation $i_1, i_2, \ldots, i_N$. For example the inversion sequence of the permutation $6, 2, 3, 4, 1, 7, 5$ is $4, 1, 1, 1, 2, 0, 0$. Here, for example, the 2 (which is the 5th element in the inversion sequence) is saying that there are exactly 2 elements in the permutation which are to the left of 5 end are greater than 5 (Yes this is true, they are: $6, 7$).

The inversion sequence $a_1, a_2, \ldots, a_N$ satisfies the conditions

$$0 \leq a_i \leq N - i \qquad \text{for} \quad i = 1, 2, \ldots, N$$

As seen there is no restriction on the elements which says $a_i = a_j$ is forbidden for $i \neq j$. This is of course very convenient for the crossover and mutation operations in GA.

Below two iterative algorithms are given. The first generates the inversion sequence of a given permutation and the second does the inverse (generates the corresponding permutation of a given inversion sequence).

**Input** $perm$ : array holding the permutation
**Output** $inv$ : array holding the inversion sequence

```
for i ← 1..N do
    { inv_i ← 0
      m ← 1
      while perm_m ≠ i do
            { if perm_m > i then inv_i ← inv_i + 1
              m ← m + 1 } }
```

**Input** $inv$ : array holding the inversion sequence
**Output** $perm$ : array holding the permutation
**Uses** $pos$ : dummy array for intermediate result[1]

```
for i ← N..1 do
    { for m ← i + 1..N do
          if pos_m ≥ inv_i + 1 then pos_m ← pos_m + 1
      pos_i ← inv_i + 1 }
for i ← 1..N do perm_{pos_i} = i
```

By this method a chromosome or a subsection of it, which has to keep a permutation will consist of a sequence of $N$ genes[2] where the allele of each element is a natural number. The maximal allele value allowed decreases by one at each element from the first position of the sequence to the last one.

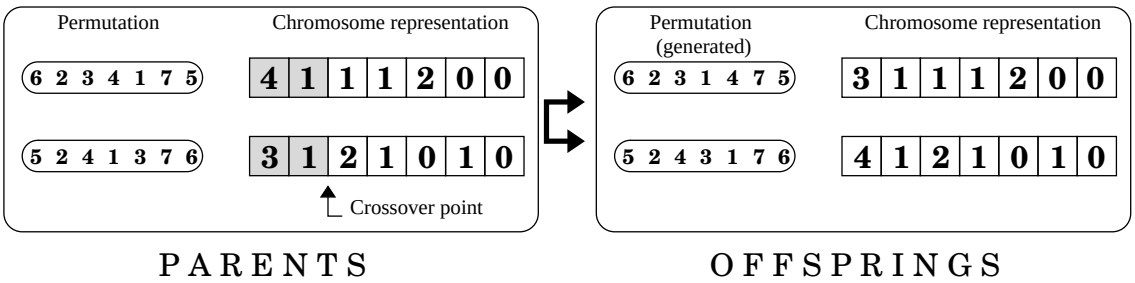
In GA applications natural number valued genes are usually represented by bit strings, which are the binary representation of that number. The limitation is very easily controlled by the choosing a restricted bit length and/or a modulo operation. Except this limitation of the maximal values, which always is the case in GA applications with numerical alleles, there is no extra restriction or order that has to be preserved throughout the GA operations. Whatever crossover or mutation will produce will inheritly correspond to a valid permutation. Now there is a question to be answered:

*What characteristics of the parents will be inherited by the offspring?*

Assuming that an ordinary bit-string one-point crossover is performed on both components of the chromosome, we can state that the offsprings will inherit characteristics from both parents. For one of the offspring, one of the parents, $p_1$, will provide the *displacement* information of some of the permutation elements (lets call them $\mathcal{E}'$) and the other parent, $p_2$ will provide a similar information for the remaining permutation elements ($\mathcal{E}''$). Of course the other offspring will receive the displacement information for $\mathcal{E}'$ and $\mathcal{E}''$ from $p_2$ and $p_1$, respectively.

Similar properties can be stated for mutation.

Below is an example coding of the two permutations $6, 2, 3, 4, 1, 7, 5$ and $5, 2, 4, 1, 3, 7, 6$ which undergo an ordinary crossover that generates two offsprings from them:




[1]The use of this array can be avoided by a more elaborated algorithm but this will not reduce the time complexity.

[2]Actually $(N-1)$ suffices, since $inv_N$ is always zero.

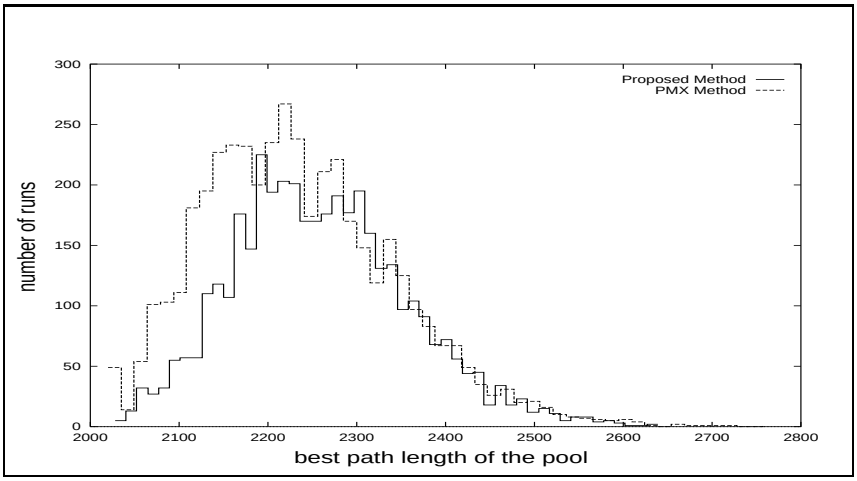
# 4 GA Solution of the TSP Problem

Both methods, PMX and the newly proposed one, are implemented as C programs. The differences are kept as local as possible. As a testbed a problem from the TSPLIB is chosen. This problem is a symmetric TSP and the data comes from a 'Real World problem' namely the road distances that connect 29 cities in Bavaria, Germany.

Both methods were run with the same settings of GA dynamics: 10-point crossover, pool size of 1000 chromosomes, 15% elitism, 0.007 mutation/gene_exchange. To avoid the differences coming from the randomization of the initial pool both methods are run 4000 times and comparison is made statistically.

The optimal distance solution of the problem is known to be 2020. Normally the average of the initial population (which is created randomly) is about $6000 - 7000$. Of course neither method converges always to the optimum but rather gets caught in local minima which are nearly optimal. Below the number distribution of the pool's bests of each of the runs for both the methods are given as histograms (dashed lines are PMX, solids are the proposed method)

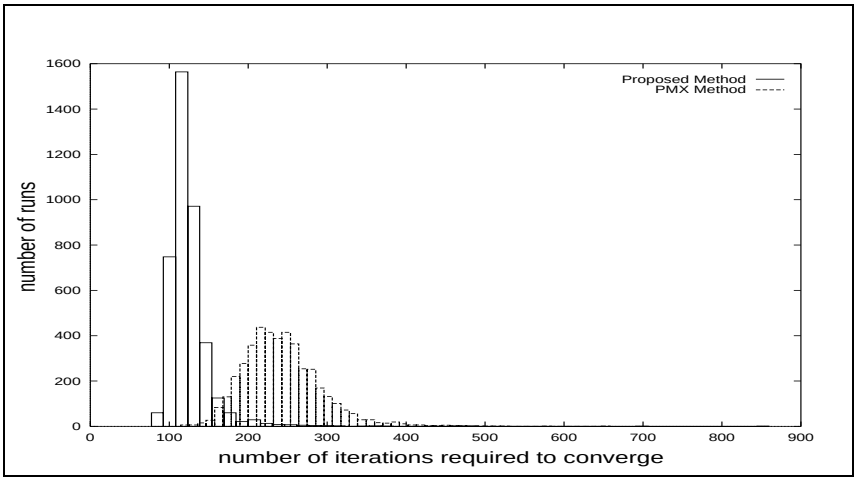
Histogram over All Runs for the Pool's Best




As observed the proposed method produces almost with the same frequency bad solutions as PMX does. The most frequently produced solution by the proposed method is better then the most frequently produced solution by PMX but is less in number by 1% over all 4000 runs. The good solutions are slightly more produced by PMX. The averages are 2259 and 2242 for the proposed method and PMS respectively. The proposed method is more stable in producing solutions, namely the standard deviation for it is 98.5 where the same figure is 108.3 for PMX.

The striking point about the proposed method is not only that it allows simple crossover and mutation, but also the high convergence rate observed in the TSP. The proposed method converges to a nearly optimal solution much more rapidly. Below the count of iterations of each run is given as a histogram over all the 4000 runs. Similar to the previous histograms the dashed lines are used for PMX and solid lines for the proposed method.

Histogram over All Runs for the Iteration Count




The count of iteration till convergence is achieved is on the average 110.5 for the proposed method. PMX requires on the average 248 iteration to converge. The stability measure of the convergence is also in favour of the new proposed method: The standard deviation for the iteration counts is 30.9 for the proposed method where it is 48.9 for PMX.

# 5 Conclusion

A new method for representing permutations as GA chromosomes has been introduced. In contrast to the conventional ones this proposed representation is not handicapped under crossover and mutation. The proposed method is used in a TSP and has proven itself as good as the conventional method. The comparative study of the results shows that the new method outperforms the conventional PMX method by a factor of 2.2 in convergence rate.

# References

[1] D.E. Goldberg and R. Lingle, Alleles, Loci, and the Travelling Salesman Problem, in:J.J. Grefenstette (ed), *Proceedings of the First International Conference on Genetic Algorithms and Their Application*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, pp. 154-159

[2] L. Davis, Applying Adaptive Algorithms to Epistatic Domains, in: *Proceedings of the International Joint Conference on Artificial Intelligence*, 1985, pp. 162-164

[3] M. Oliver, D.J. Smith and J.R.C. Holland, A Study of Permutation Crossover Operators on the Travelling Salesman Problem, in:J.J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, pp. 224-230

[4] M. Hall, *American Mathematical Society, Proceedings of the Symposium on Pure Mathematics*, 1963, **6** pp. 203

[5] S. Even, *Algorithmic Combinatorics*, The Macmillan Company, NY, 1973