ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 8, No. 1, 2013, *pp.* 055-062

Shared Crossover Method for Solving Traveling Salesman Problem

Mouhammd Al kasassbeh, Ahmad Alabadleh, Tahsen Al-Ramadeen

[1] Information Technology Department, Mutah University, Karak, Jordan

(*Received September 11, 2012, accepted November 30, 2012*)

Abstract. Genetic algorithms (GA) are evolutionary techniques that used crossover and mutation operators to solve optimization problems using a survival of the fittest idea. They have been used successfully in a variety of different problems, including the traveling salesman problem. The main idea of Traveling Salesman Problem (TSP) is to find the minimum traveling cost for visiting cities; the salesman must visit each city exactly once and return to the starting point of origin. Genetic algorithms are search methods that employ processes found in natural biological evolution. These algorithms search on a given population of potential solutions to find those that pass some specifications or criteria. In this paper, we apply modified genetic algorithm methodology for finding near-optimal solutions for TSP problem using shared neighbours to insure that the closest cities to have the highest priorities to be carried out to the next generation.

Keywords: *Traveling Salesman Problem (TSP), Genetic Algorithm (GA), Order Crossover (OX), Swap Crossover, Shared Crossover.*

1. Introduction

1.1. Traveling Salesman Problem (TSP)

The main idea of the TSP is the discovery of the shortest possible tour path through a given set of nodes or cities. The comprehensive surveys of works on the TSP can be found in [1-6]. Researchers have suggested genetic algorithms (GA's) for solving TSP [7]. The TSP is one of the significant subjects, which has been widely addressed by mathematicians and computer scientists. Its importance stems from the fact, there is a plenty of fields in which finds potential applications such as DNA fragment assembly and VLSI design. Formally, the TSP may be defined as follows [8].

It is a combination problem with the objective of finding the path of the shortest length (or the minimum cost) on an undirected graph that represents cities or nodes to be visited. The traveling salesman begins at one node, visits all other nodes consecutively only once, and finally returns to the starting point. In other words, given n cities, named {c1, c2, …, cn}, and permutations, {σ1, σ2, …, σn}, the goal is to choose σi such that the sum of all Euclidean distances between each node and its successor is minimized. The successor of the last node in the permutation is the first one. The Euclidean distance d between any two cities with coordinate (x1, y1) and (x2, y2) is calculated by.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

And the minimum total distance is calculated as follows;

$$Minimize\left[\sum_{i=1}^{N-1} d\left(c_i, c_{i+1}\right) + d\left(c_N, c_1\right)\right] \tag{2}$$

Recently, Robert Bosch [9] created a 100,000-city instance of the traveling salesman problem (TSP) that provides a representation of Leonardo da Vinci's Mona Lisa as a continuous-line drawing. An optimal solution for this case set a new world record for the TSP. The current best known results for the Mona Lisa TSP are; Tour: 5,757,191, Bound: 5,757,044 and Gap 147 (0.0026%). The tour was found on March 17, 2009, by Yuichi Nagata [9]. The bound ***B*** is given such that no tour has length less than B; this bound was found on November 4, 2009, with the Concorde code [9].

The remainder of this paper is organized as follows: in Section 2, we discuss related work .Section 3 the problem statement. In Section 4 provides an overview of our new crossover technique to solve the TSP. The discussion and the comparison between OX, swap and our proposed shared crossover are illustrated in Section 5. Finally, we summarize and conclude in Section 6.

1.2. Genetic Algorithms (GA)

Genetic Algorithms are search algorithms developed by Johan Holland in the 1970's [5], it can be used to solve a variety of problems that are not easy to solve using other techniques. Fundamentally, Genetic Algorithms consist of the three basic operations, selection operation, crossover operation, and mutation operation. When we use a GA to solve a problem, random possible solutions will be generated, each one of them will be tested until we have a best solution, then it will be good enough for a specific problem.

Selection operation used to select a random possible solution from a set of population called chromosomes using fitness function, which is determined by the type of the problem. The crossover operation used to produce new chromosomes called children chromosomes from the parent chromosomes by choose a randomly crossover point. Finally, the mutation used to make some change on the children chromosomes to make the solution very close to the realty. Figure 1 illustrates the simple GA structure.

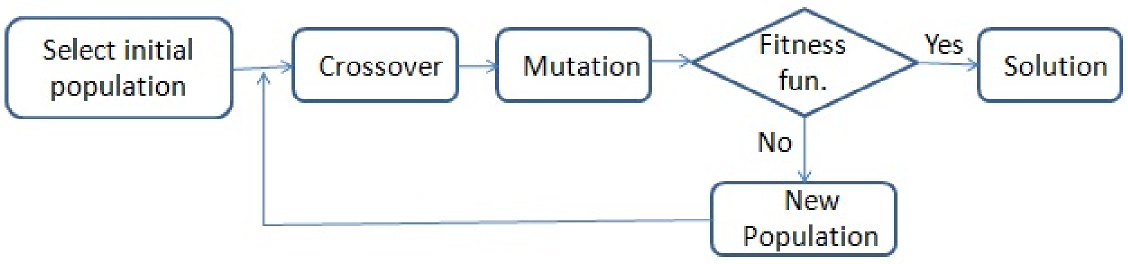

Figure 1: The structure of a simple genetic algorithm

Moreover, to illustrate GA operations, here is a very simple example:
Suppose we have the following population:

No	Chromosome
1	1001100011
2	1110001111
3	1010101011
4	1100110011

- Selection : suppose we select chromosome 1 and 3

- Crossover: suppose the crossover point is 4

  Parent 1: 1001 100011 Child 1 : 1001101011

  Parent 2: 1010 101011 Child 2 : 1010100011

- Mutation : Usually done with small probability

Child 1: 1 0 0 0 1 0 1 0 1 1 Child 2: 1 0 1 0 1 0 0 0 0 1

In this example, we gave a brief idea about the GA and its operations, in the following section. We will talk about the previous work that has been done in the area of GA and TSP.

2. Related Work

Previously, many researchers had proposed and used many algorithms and enhancement approaches for solving the TSP. For example, a research by [10] resolved the problem of crossover in genetic algorithm using DNA-strands. Although the crossover technique might be different from one problem to another depending on the problem itself, they show that it is possible to find a suitable crossover for NP-Complete problems such as the TSP . In brief, they represent a new approach to the simulation of genetic algorithms with DNA-strands. Similarly, the work of [11] used "2-opt" algorithm for local optimization, which is a simple local search algorithm, the main idea is to take a route that crosses over itself and reorders it again, also a new breeding technique was used that involves selecting two chromosomes that have shared partial tours. Furthermore, the work of [12] used one-chromosome generation in their algorithm, they rely on mutation without applying crossover, and they used 3-opt for local optimization, 3-opt analysis involves deleting three edges in a tour, later, they were trying reconnecting the graph in all other possible ways, then they evaluate each reconnection method to find the optimum solution. This process is then repeated for a different set of three connections. In [13] they used Lin-Kernighan algorithm for local optimization, which involves swapping pairs of sub-tours to make a new tour. It is a generalization of 2-opt and 3-opt. the 2-opt and 3-opt work by switching two or three paths to make the tour shorter. Lin-Kernighan is adaptive and at each step decides how many paths between cities need to be switched to find a shorter tour. What is more in the area of GA and TSP, the work of [14] claims that the use of the heuristic to solve the TSP gives quite well results when they tested on a set of 41 standard problems with known optimal objective values, and they find the optimal solution in the majority of the cases. Their procedure, mainly, combines a local tour enhancement heuristic into a random-key genetic algorithm and shows good results in both solution quality and computation time.

The previous algorithms used local optimization algorithm to solve the problem, which leads to finding a local optimal solution. These algorithms, in addition, work with high time complexities. New crossover and mutation techniques were developed, but they fail with a large number of cities, and these techniques need large memory space [15].

3. Problem Statement

The majority of the existing crossover methods generally find only the local optimal solutions in TSP. Usually the crossover methods show degradation in performance when they applied using the large number of cities in TSP [16, 17]. In addition, these methods are not efficient in terms of memory and performance. Typically, crossover point is determined randomly. In TSP, we must guarantee that no city will appear in the child chromosome more than one time after crossover the operation.

4. The Proposed Procedure

Our study aims to find a new crossover technique to solve the TSP, and it should be genetic, simple and fast at the same time. Furthermore, we concentrated in our study on reducing the execution time and number of populations. We proposed a new procedure for the crossover operation in solving TSP using GA; the detailed clarifications about how it works are as follows;

1. Pick two chromosomes; one of them is the fittest according to the fitness function. i.e .one of them has the shortest path found so far.
2. Randomly, we chose an initial city from the fittest chromosome to start from.
3. Find the exact location of this selected city (current city) in both of the two parent chromosomes.
4. Determine the neighbor cities of the current city in the two parent chromosomes.

5. Pick the shared neighbor between the two parent chromosomes and insert it in the new child chromosome.
6. If the shared neighbor which has been selected was visited before, or if the current city has no shared neighbor between the two parent chromosomes, pick up the nearest neighbor to the current city.
7. Repeat steps 3-6 until you reach the last not visited city, and then insert it in the new child chromosome.

Further down, our algorithm has been explained in details;

```
Function SharedCrossover

    C1 :Best chromosome
    C2 :Random Chromosome
    InitialCity :Random city from C1

    Neighbors[0..3]:Contain the Neighbors' cities around InitialCity in C1 and C2 and its distances

    i=0
    While not-end-cities

    If there is a shared city in Neighbors [] and NOT(SharedCity .Visited)
        ChildChrom[i].City =SharedCity
        SharedCity .visited =true
        InitialCity =SharedCity
    Else
        NearestCity =Neighbors.City (Min (Neighbors.Distance )) and NOT(NearestCity.Visted)
        ChildChrom[i].City =LowestCity
        NearestCity.Visited =true
        InitialCity =NearestCity
    End if
    i++
    End while
End Function
```

An example, to give the reader clear idea, how our proposed algorithm works. At first, the data for our example is given below;

City	London	Oxford	Cambridge	Brighton	Bath
Code	1	2	3	4	5

The second table shows the distance between the cities in miles;

	London	Oxford	Cambridge	Brighton	Bath
London	0	350	50	280	470
Oxford		0	130	270	310
Cambridge			0	210	340
Brighton				0	220
Bath					0

According to the above algorithm, the best chromosome is as follow, with the minimum fitness value 790 miles.

C1(Best)	1	3	4	5	2

Subsequently, we choose randomly the other chromosome C2

C2(Random)	1	2	3	4	5

After applying the proposed algorithm, the following table illustrates the algorithm tracing;

	InitialCity	*InitialCity neighbors in C1*		*InitialCity neighbors in C2*		*SharedCity*	*NearestCity if no shared*
Step 1	(3) Randomly selected in the first time from C1	1	4	2	4	4 Add to child	No need
Step 2	4	3	5	3	5	5 Add to child (3 Already visited)	No need
Step 3	5	4	2	4		4 (But it was visited before)	2 Add to child
Step 4	2	5		1	3	No shared	1 Add to child
Step 5	1	All visited					

In step 1, the initial city was picked up randomly. In our example, we selected Cambridge city with label number 3. As we can see in chromosome c1, the neighbors of city 3 are city 2 and city 4. Likewise, the neighbors of city 3 in chromosome c2 are city 4 and city 5. Apparently, city 4 is the shared city between chromosome c1 and c2, subsequently; it will be added to the child chromosome, and it will be the new initial city as shown at step 2. The same process, we did on city 3, will be repeated on city 4 again, and so on. In case there is no shared city between chromosome c1 and c2, or the shared cities have been previously visited, the closest in distance will be chosen as shown in step 4. The table below will show the output of the above steps;

Step 1 output	3				
Step 2 output	3	4			
Step 3 output	3	4	5		
Step 4 output	3	4	5	2	
Step 5(Final Child Chromosome)	3	4	5	2	1

In the final step (step 5), the first child chromosome is created with values;

first child chromosome	3	4	5	2	1

This child will be carried out to the next population and so on.

5. Experimental Results

All the tested methods we adopted in our work, including our proposed algorithm are implemented using C++ programing language. The suggested algorithm was tested with a crossover probability of 90% and a mutation probability of 10% [18, 19]. The standard population size was set at 100 chromosomes, and each chromosome has 29 cities, the testdata has been chosen from the TSPLIB. The data obtained from a real world problem namely, the road distances that contained 29 cities in Bavaria, Germany(street distance), the source of our data is Zuse Institute Berlin [20].

The experiment was carried out using three crossover techniques; OX, swap, and our proposed shared crossover. In this experiment, we tried to show the performance of our technique in two main cases comparing to the OX and the swap techniques. The first case, is to find the optimal solution at specific generation; in this case, the number of generations was 1000, when we tested the three algorithms the results were as follows;

- The optimal solution of the proposed shared crossover during the 1000 generation was 3720, precisely at 640 generations.
- For the OX algorithm, it reaches the optimal solution 3870 in 660 generations.
- Lastly, the swap reaches the optimal solution 4000 in 870 generations.

From the results of the first case, it is noticeable that the shared crossover reaches the best solution in fewer numbers of generations, as it can be observed in Figure 2.

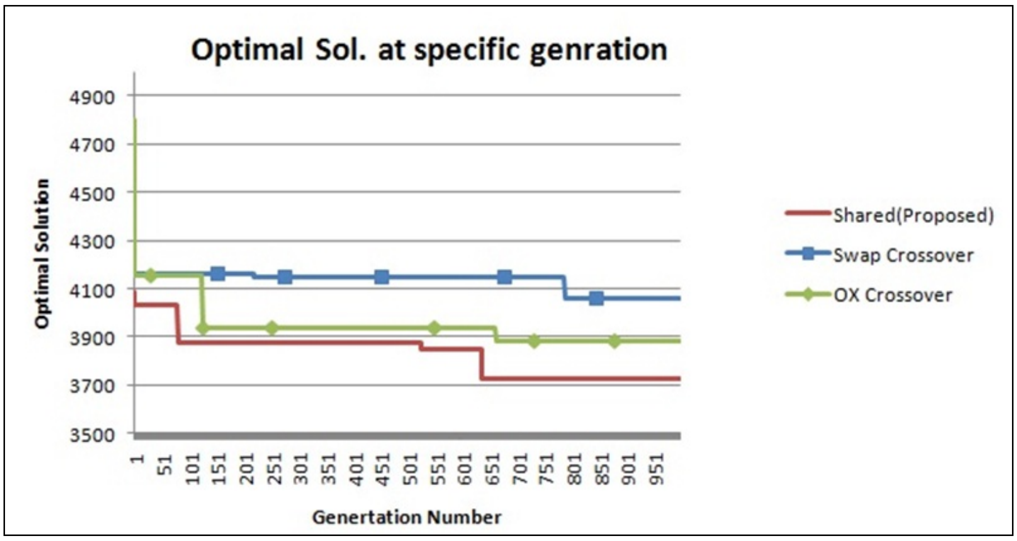

Figure 2: Optimal Solution at specific generation

The second case was to find the number of generations that they used in their process when we gave them a specific fitness value. The results are as follow:

The fitness value	**Number of generation in the shared crossover**	**Number of generation in the OX**	**Number of generation in the Swap**
4400	50	60	103
4300	55	70	136
4200	55	83	560
4100	213	300	1030
4000	733	860	1320
3900	*912*	1101	1473
3800	1000	1224	1683
3700	1511	1570	2517
3600	2120	2214	5000
3500	2431	2580	6000

Figure 3 represents the graph of these results. The number of the generations in the shared crossover algorithm always has fewer numbers of generations than the others to reach the optimal solution; therefore, the execution time will be less.

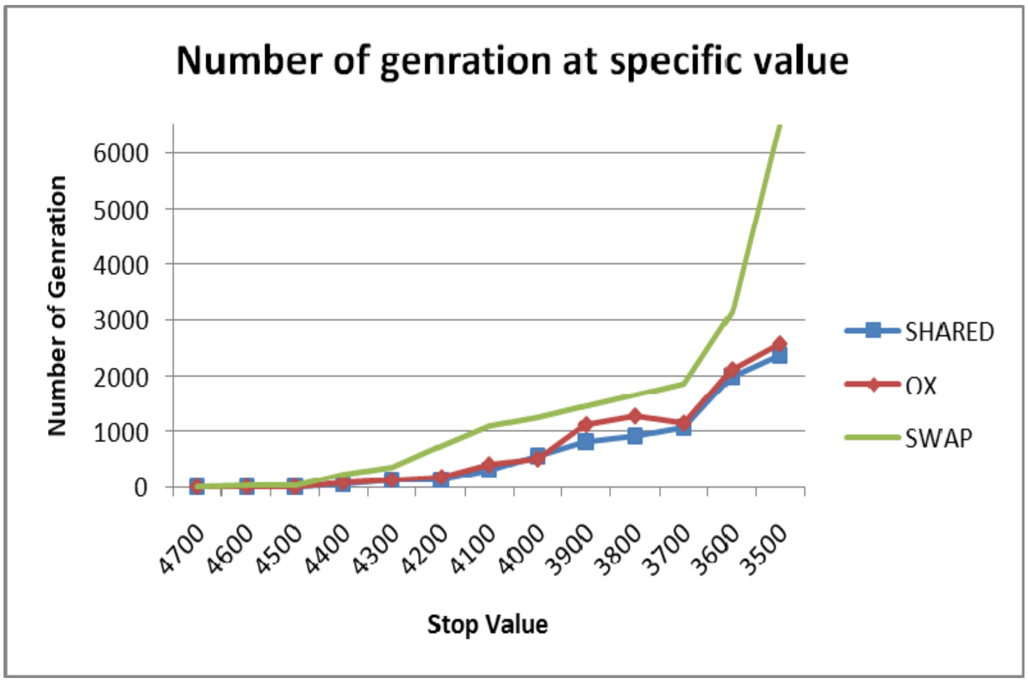

Figure 3: Number of generation at specific value

Finally, the simulation's results indicate that, the proposed crossover algorithm supersedes the swap and the OX algorithms, in terms of the number of generations and execution time.

6. Conclusion

The proposed GA shared crossover technique was developed to assist for obtaining the optimal tour to visit number of cities. The new technique depends on passing as many as possible of the shared paths between cities to the next generation. With a guarantee that none of the cities will appear in the child chromosome more than once after applying the crossover method, and we achieved this goal without using any time consuming search methods. We have shown in the experimental results a great reduction in the execution time.

7. References

[1] J. J. Bentley, "Fast algorithms for geometric traveling salesman problems," *INFORMS Journal on Computing*, vol. 4, pp. 387, 1992.

[2] D. B. Fogel, "Empirical estimation of the computation required to discover approximate solutions to the traveling salesman problem using evolutionary programming," 1993.

[3] G. G. Mitchell, D. O'Donoghue, D. Barnes, and M. McCarville, "GeneRepair-a repair operator for genetic algorithms." 2003.

[4] G. G. Mitchell, D. O'Donoghue, and A. Trenaman, "A new operator for efficient evolutionary solutions to the travelling salesman problem," 2000.

[5] M. Mitchell, *An introduction to genetic algorithms*: The MIT press, 1998.

[6] R. Yang, "Solving large travelling salesman problems with small populations," 1997.

[7] C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, pp. 129–170, 1999.

[8] S. Sur-Kolay, S. Banerjee, and C. A. Murthy, "Flavours of Traveling Salesman Problem in VLSI Design," 2003.

[9] R. Bosch, "Mona Lisa TSP Problem," Georgia, 2009, pp. The Traveling Salesman Problem. 2010.

[10] G. Moreno, "Solving Travelling Salesman Problem in a Simulation of Genetic Algorithms with DNA," 2008.

[11] R. M. Brady, "Optimization strategies gleaned from biological evolution," 1985.

[12] O. Martin, S. W. Otto, and E. W. Felten, "Large-step Markov chains for the TSP incorporating local search heuristics," *Operations Research Letters*, vol. 11, pp. 219-224, 1992.

[13] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," 1996.

[14] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, pp. 38-53, 2006.

[15] X. B. Hu and E. Di Paolo, "A Hybrid Genetic Algorithm for the Travelling Salesman Problem," *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, pp. 357-367, 2008.

[16] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, pp. 92-99, 2007.

[17] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," 2002.

[18] J. Zhang, H. S. H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 326-335, 2007.

[19] L. I. Geng, Z. O. U. Jin, and Z. Bing, "The Genetic Algorithm Simulated Annealing of Large Probability of Mutation and its Application in Reservoir Optimization," *China Rural Water and Hydropower*, 2010.

[20] Z. I. Berlin, "MP-TESTDATA- the TSPLIB Symmetric Traveling Salesman Problem Instances," vol. 2010: Zuse Institute Berlin, 2010.