An Efficient Solution to Travelling Salesman Problem using Genetic Algorithm with Modified Crossover Operator

Md. Sabir Hossain
Chittagong University of Engineering & Technology
24 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Muhammad Nomani Kabir
Universiti Malaysia Pahang
84 PUBLICATIONS **480** CITATIONS
SEE PROFILE

Mohammad Mainul Islam
University of Houston
5 PUBLICATIONS **11** CITATIONS
SEE PROFILE

EMITTER International Journal of Engineering Technology
ISSN: 2443-1168, Vol. 7, No. 2, December 2019, pp. 480~493
DOI : 10.24003/emitter.v7i2.380

An Efficient Solution to Travelling Salesman Problem using Genetic Algorithm with a Modified Crossover Operator

Md. Sabir Hossain[1*], Sadman Sakib Choudhury[1], S. M. Afif Ibne Hayat[1], Ahsan Sadee Tanim[2], Muhammad Nomani Kabir[3], Mohammad Mainul Islam[4]

[1]Chittagong University of Engineering & Technology, Chittagong, Bangladesh
[2]The International University of Scholars, Dhaka, Bangladesh
[3]Universiti Malaysia Pahang, Malaysia
[4]Verizon Media, California, USA
[*]Email: sabir.cse@cuet.ac.bd

Received March 30, 2019; Revised May 28, 2019; Accepted June 30, 2019

Abstract

Traveling salesman problem (TSP) is a famous NP-hard problem in the area of combinatorial optimization. It is utilized to locate the shortest possible route that visits every city precisely once and comes back to the beginning point from a given set of cities and distances. This paper presents an efficient and effective solution for solving such a problem. A modified crossover method using Minimal Weight Variable Order Selection Crossover (MWVOSX) operator, a modified mutation using local optimization and a modified selection method using KMST is proposed. MWVOSX operator chooses a particular order from multiple orders which have the minimum cost and takes the remaining from the other parent in backward and forward order. Then it creates two new offspring. Furthermore, it selects the least weight from the two offspring. The efficiency of the proposed algorithm is compared to the classical genetic algorithm. Comparisons show that the proposed algorithm provides much efficient results than the existing classical genetic algorithm.

Keywords: Traveling Salesman Problem, Crossover Operator, Minimal Weight Variable, Genetic Algorithm.

1. INTRODUCTION

For a given set of nodes and the travel distance between each node-pair, TSP is to find the most ideal way of visiting every node such that the total traveling distance is minimized. TSP is an NP-Hard problem, which implies that no recognized algorithm can resolve it in polynomial time. The traveling salesman problem can be solved by using brute force, DP approach, genetic algorithms, ant colony optimization, simulation annealing, etc. It is an old problem and there is no perfect solution to find the optimal route for an unlimited number of separated nodes.

The simplest way to solve TSP is to try all the possible routes to visit all the cities while a particular route combination is found such that summing distances between the cities is the shortest. This very straight-forward solution is called a brute-force or exhaustive search. However, this technique has a shortcoming of immense importance – running time of Θ((n-1)!) Permutation of n cities and calculate the total distance to visit all the cities. And as n grows, the factorial (n-1)! rapidly rises to a larger number than all polynomials and exponential features (however, lower than double exponential functions) n-1. This enormous growth of possible routes means enormous growth of time needed for solving TSP even for contemporary computers. On the other side, there is no doubt that the result will be surely the shortest route. Unfortunately, due to its time complexity Θ(N!), this algorithm is not suitable for this work. If there exists another algorithm with better time complexity and exactness (thus returning the routes of the same quality as with brute-force algorithm, i.e. optimal route), it should be chosen for the implementation.

The main objective of this research is to find a solution that is efficient than the naive and dynamic programming approach. Using Modifying mutation process and crossover using MWVOSX operator, as mutation and crossover process is the most crucial part in a genetic algorithm which provides lower time complexity than the other approaches.

The remaining of this paper is arranged in the following categories: Section 2 contains some overview of related work that was done in previous; Section 3 discusses the contribution of the current research. Section 4 shows the proposed methodology in details. Finally, Section 4 is about processing example and rest 5 and 6 are summarizing the evaluation and future work.

2. RELATED WORKS

A lot of research work done in TSP earlier. In paper [1] a Discrete Symbiotic Organisms Search (DSOS) is proposed. To demonstrate that the proposed arrangement approach of the DSOS is a promising system for taking care of combinatorial issues like the TSPs, a lot of benchmarks of symmetric TSP occasions chose from the TSPLIB library are utilized to assess its execution against other heuristic calculations. Another work on various parent selection methods named as Elitism, Roulette Wheel & amp; Tournament selection methods to solve TSP [2]. For a small number of inputs, they show a similar result, but they produce better results for larger inputs.

Paper [3] proposed a discrete imperialist competitive algorithm to solve TSP. It is a socio-politically driven meta-heuristic procedure. They introduced the 2-opt algorithm in the revolution procedure. It shows excellent performance in a small-scale dataset. Other authors propose a quantum technique to resolve TSP [4] utilizing a phase evaluation strategy. They apply a phase estimation algorithm for finding distances of all the routes. Then calculate the least possible distance to find the route. Paper [5] TSP is solved by Black Hole algorithm. BH algorithm is based on the meta-heuristic algorithm. This algorithm is faster and provides a good solution than a generic algorithm. Six different crossover procedures are described in [6]. The algorithms that are used are Deterministic algorithm, Approximation algorithm, Genetic algorithm and applied the genetic algorithm. Six crossovers which were discussed in this journal are Uniform crossover operator, Cycle Crossover, Partially Mapped Crossover (PMX), the uniform partially mapped crossover (UPMX) and Ordered Crossover (OX). OX performed the best in their experiment. A new method called "Modified Cycle Crossover (CX2)" operator [7] which is used to solve TSP using a genetic algorithm. There are 6 steps to follow. Then PMX and OX are compared to the proposed method. CX2 performed better compared to the other two in the experiment. In [8], Varshika Dwivedi, Tarun Chauhan, Sanu Saxena, Princie Agrawal proposed a solution using genetic algorithm operators which is Sequential Constructive Crossover (SCX) operator. They show SCX result in a high-quality solution. Also, they provide a relative comparison between genetic algorithm and dynamic programming for solving the problem. Md. Lutful Islam, Danish Pandhare, Arshad Makhthedar, Nadeem Shaikh [9] work on Partially Matched Crossover, Two Point Crossover and order Crossover are used in this work. They show enough efficiency for a high number of inputs. Map framework and a parallel algorithm are mentioned in this paper. In recent days TSP is solved by artificial bee colony (ABC) [10]. The author used four phases named - Initialization phase, a phase of employed bees, a phase of onlooker bees, a phase of scout bee. In every phase, the system memorizes the best solution and evaluated it.

The authors in [11] discussed solving TSP using an artificial ant colony system (ACS). They presented combinatoric optimization, candidate list methods to solve the problem. Their key contribution is that ACS can give better performance in solving TSP than any other methods in almost every case. There are many ways in which ACS can be improved so the time needed to complete tours.

3. ORIGINALITY

In this noble research, we proposed a modified genetic algorithm approach to solve the traveling salesman problem. In the algorithm selection process will be modified using Kruskal's minimum spanning tree algorithm and the mutation process will be modified using local optimization. For a weighted, undirected and connected graph a minimal spanning tree is a spanning tree with weight less or equal to every other spanning tree. From a set of local solutions, finding an optimal solution is a local optimization. However, we were unable to modify the selection process using KMST. MWVOSX crossover operator was used in our proposed genetic algorithm method. Traveling salesman problem has several applications in real-world phenomena. Such as in complex optimization problems, shift scheduling, planning, airline crew scheduling, etc.

4. SYSTEM DESIGN

In this proposed method, several steps have been done including distance calculation using fitness function, modified crossover using MWVOSX operator.

4.1 Dynamic Programming

In this method, the tour begins at node C_1. The minimum cost of the cycle from C_1 to some other node C_i (visiting each node exactly once and then back to c_1) can be said as $cost_{1i}$ + dis_{i1} where $cost_{1i}$ is the cost of the shortest path from C_1 to C_i. $cost_{1i}$ can be calculated with dynamic programming by considering all subsets. Let D(S, C_i) be the shortest path from C_1 to C_i visiting each node exactly once.

If |S| = 2, then D(S, C_i) = dis_{1i}. If |S| > 2 then the D(S, C_i) is specified by the recurrence min(D(S −{C_i}, j) + dis_{ji}).

The complexity of this algorithm is n^2*2^n. Although improved than the brute-force approach, this algorithm becomes impractical and cannot be used for larger problems.

4.2 Fitness Function

A fitness function or objective function in a genetic algorithm for traveling salesman problem means the probability to be selected for mating is proportional to the value of fitness function. Fitness function evaluation is fused to allots a value to every organism, noted as $\boldsymbol{f_i}$. This $\boldsymbol{f_i}$ value is a figure of legitimacy which is determined by utilizing any domain knowledge that applies. On a fundamental level, this is the main point in the algorithm that domain knowledge is important. Organisms are picked utilizing the fitness value as a guide, where individuals with higher fitness values are picked more often.

For minimization problem, one method for characterizing a fitness function is as $\boldsymbol{f_i}$ is the objective function. Since TSP stands as a minimization problem; we consider this fitness function, where $\boldsymbol{f_i}$ determines Euclidean distance of every tour of a population, where tours are represented by chromosomes. The technique utilized here was to compute the total Euclidean distance $\boldsymbol{D_i}$ for every population first, then calculate $\boldsymbol{f_i}$ by utilizing the equation,

$$f_i = 1 / D_i$$

where D_i is the Euclidean distance of tours in population.

4.3 Generic Algorithm

A genetic algorithm has a place with the larger class of evolutionary algorithms. It is inspired by the theory of natural selection by Charles Darwin. Genetic algorithms rely on bio-inspired operators like mutation, selection, and crossover. It is hugely used to generate high-quality solutions to search problems and optimization problems such as integer nonlinear problems (INLP). The typical genetic algorithm requires two things. They are a genetic depiction of solution dominion and fitness function for evaluation of solution dominion. Genetic algorithms are broadly utilized in numerous fields such as automation, automotive design, improved telecommunications routing, engineering plan, and computer-aided molecular plan.

```
1.  for i: = 1 to population size [Input of cities as population]
2.  start
3.     select new starting city c;
4.     tour[i] := adjacent city (starting city = c);
5.     optimize-local (tour[i])
6.  end;
7.  while evolution do
8.  start
9.     select parent_l and parent_2;
              [Selection]
10. child := crossover (parent_l, parent_2) ;
              [Crossover]
11. activate-edges (child, parent_l);
12. optimize-local (child);                    [Mutation]
13. if ¬∃i≤population size | length(tour{i]) - length(child) |≤ a [Survival
    of the fittest]
14. ^ ∃i≤population size length(tour{i]) ≥ length(child)
15. then substitute the longest tour by child
16. end;
```

Figure 1. Pseudocode of generic algorithm

To introduce the stream of a capacity of a program or framework through a graph or outline is called a Flowchart. A flowchart is the graphical introduction type of the arrangement of an issue.

Genetic algorithm (GA) is an adaptable optimization technique. Figure 2 demonstrates the optimization procedures of GA, the two essential operations are crossover and mutation. The GA associates the best of the last generation through the crossover, in which parameter esteems are traded between parents to form a child. A portion of the parameters mutate.

The objective function at that point decides on the fitness of the new arrangements of parameters and the calculation repeats until the point when it converges. With these two operators, the GA can investigate the full cost surface to abstain from falling into local minima. In the meantime, it exploits the best highlights of the previous generation to converge to progressively improved parameter groups.

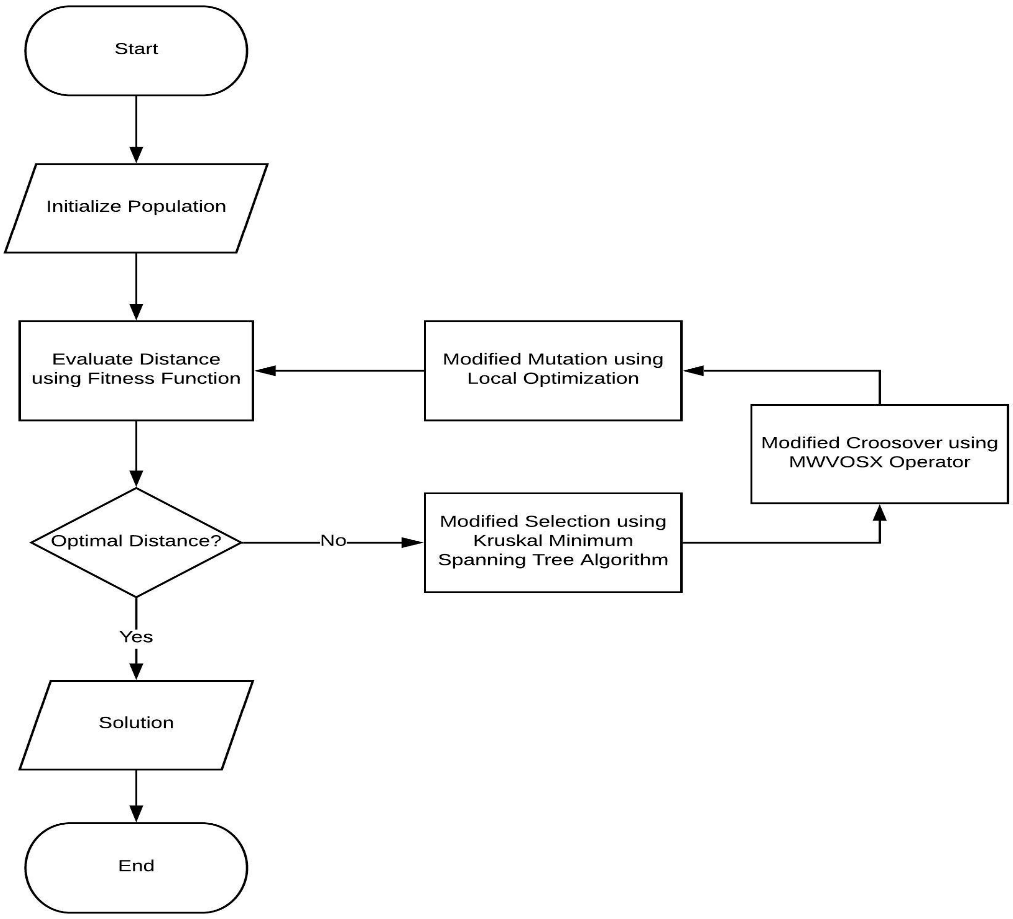

Start
Initialize Population
Evaluate Distance using Fitness Function
Modified Mutation using Local Optimization
Modified Croosover using MWVOSX Operator
Optimal Distance?
No
Modified Selection using Kruskal Minimum Spanning Tree Algorithm
Yes
Solution
End

Figure 2. Overview of Proposed Algorithm.

```
1. for j := 1 to #city do      {#city = number of cities}
2.  for i := 1 to #pop do      {#pop = population size}    [Generation of
   the city-populations]
3.  start
4.     choose new starting city c;
5.     tour[j,i] := adjacent city (starting city = c);
6. optimize_local (tour[j,i])
7.  end;
8.  while evolution := true do
9.  start
```

```
10.     for j := 1 to #city do
11.      for k := 1 to #N do      {#N = number of steps until
    degeneration}
12.      start
13.          choose parent_1 and parent_2;
                      [Selection]
14.          optimize-KMST (tour[j,i])     {KMST = Kruskal Minimum
    Spanning Tree}
15. child := crossover_MWVOSX (parent_1, parent_2) ;
             [Crossover]
16.          activate-edges (child, parent_1);
17.          optimize-local (child);
                      [Mutation]
18. if  ¬∃i≤ population size | length(tour{i]) - length(child) | ≤ a
    [Survival of the fittest]
19. ^ ∃i≤ population size length (tour{i]) ≥ length(child)
20. then substitute the longest tour on city j by child
21. end;
22. for k := 1 to #m do      {#m = number of merging steps}
    [Refreshing]
23. start
24. choose arbitrarily i, jand i’, j’ with tour [i,j] new on city i’ and tour
    [i',j’] new on city i;
25. swap tour [i,j] and tour [i', j’]
26. end; end;
```

Figure 3. Pseudocode of proposed algorithm

5. PROCESSING EXAMPLE

In a traditional traveling salesman problem, a salesman has to start from one city and visit all another city exactly once and come back to starting city. TSP is all about the least distance path with above conditions. Let's consider a problem where there are 8 cities numbered from 1 to 8.

1	2	3	4	5	6	7	8

Step 1:
This step starts with randomly generated k chromosomes, called population. Each population represents an individual solution. For each chromosome, the starting city is fixed.

1	2	7	6	8	5	3	4

Chromosome 1

1	5	3	8	4	6	2	7

Chromosome 2

1	5	8	7	2	6	4	3

Chromosome 3

1	8	3	6	5	7	4	2

Chromosome 4

Step 2:
Using a selection method, two chromosomes are selected as parents. For consideration, let's take parent 1 as chromosome 4 and parent 2 as chromosome 1.

1	8	3	6	5	7	4	2

Parent 1

1	2	7	6	8	5	3	4

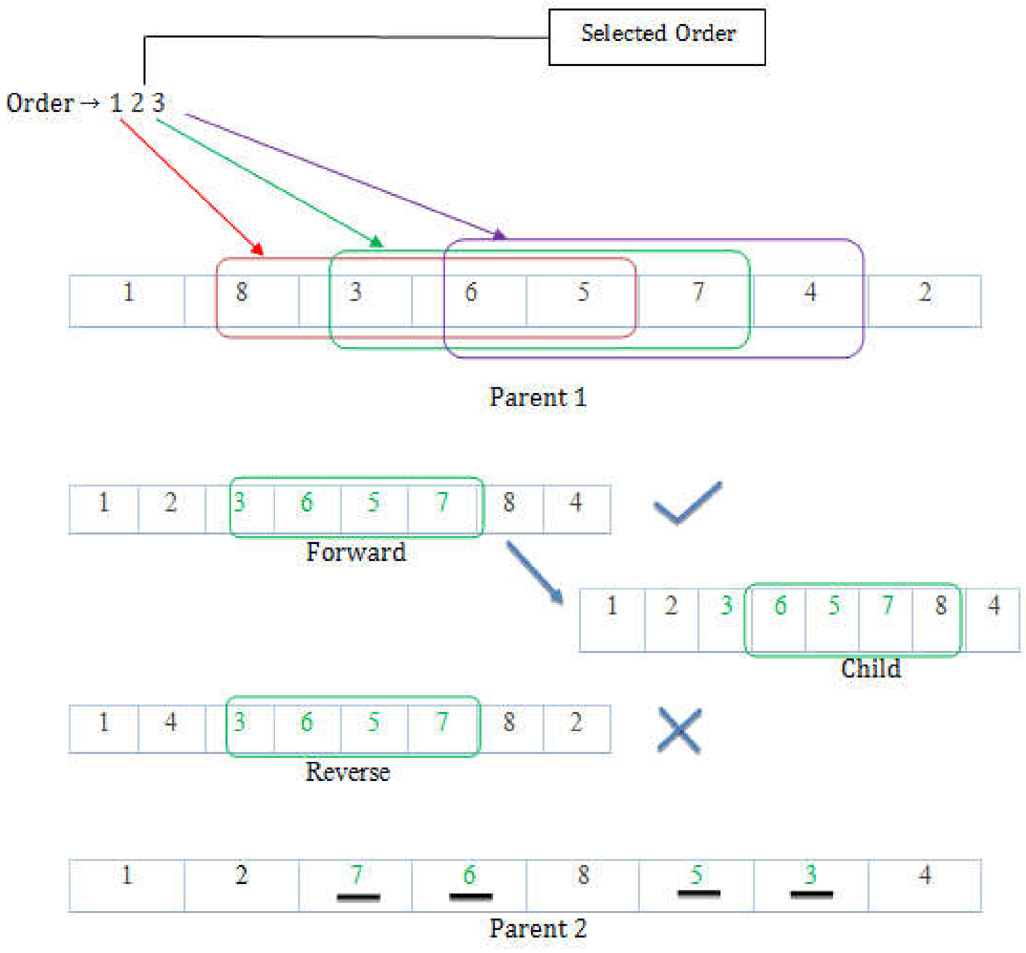
Parent 2

Step 3:
Firstly, a number of multiple swaths with consecutive cities in specific points are taken for the following step from parent 1.

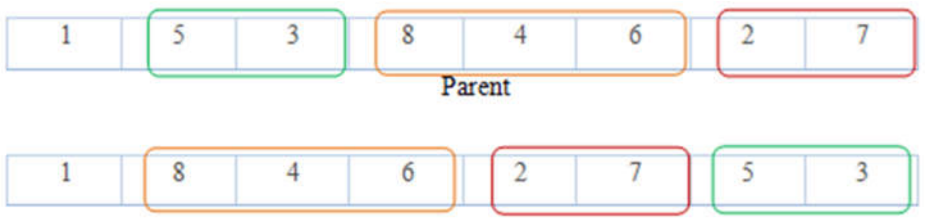
Step 4:
The length is calculated for each selected swath using order crossover and the swath with minimal weight is selected and taken as child chromosome.

Step 5:
We create two duplicates of the child as child_1(Forward)& child_2(Reverse), and then forward them to fill the remaining from parent_2 in forward and backward order.

Step 6:
The total distance of child_1 & child_2 is calculated. The smallest weighted child is selected.



Step 7:
When the offsprings are almost identical with each other and the total distance is not converging any more then the mutation is done. Mutation is done in a randomly selected parent chromosome.



Step 8:
All the offsprings or children are evaluated through a fitness function to find the optimal tour for the salesman.

6. EXPERIMENTAL RESULTS AND ANALYSIS

To analyze the proposed algorithm, it was tested for 10 different data sets consisting of 5 to 50 cities. We can analyze the algorithm on criteria which is based on the generation number to find the optimal distance. The generation number can vary for the same dataset to find an optimal solution. So, we considered an average generation number.

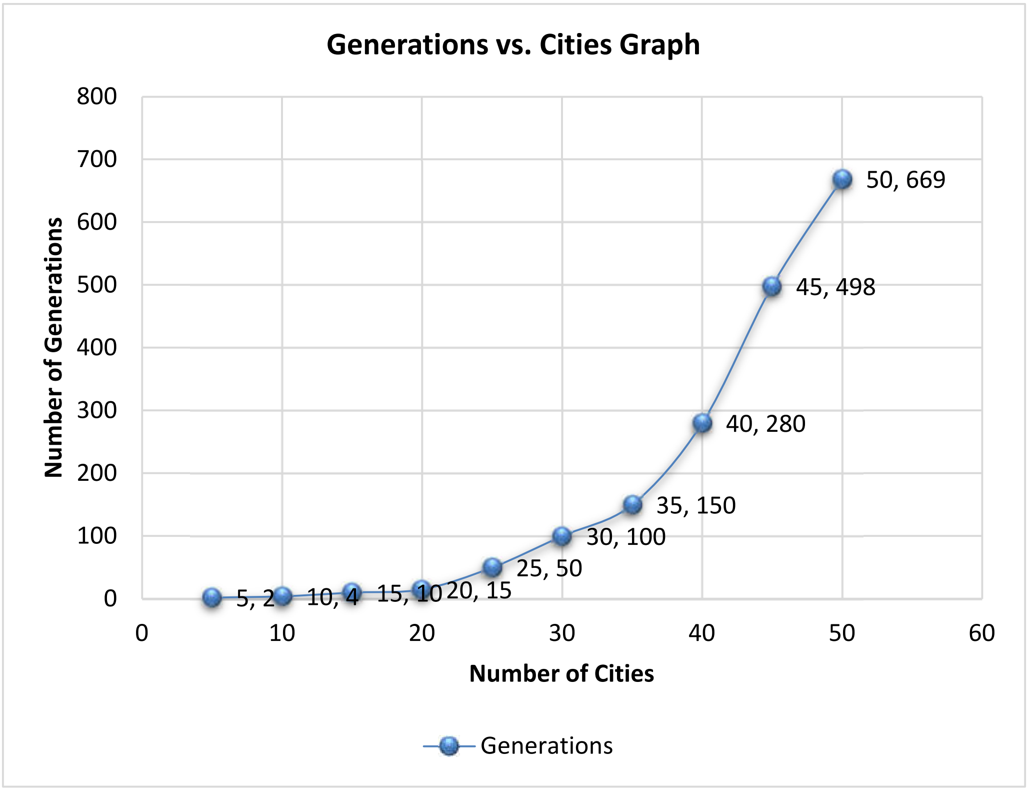


Figure 4. Generations vs. Cities Graph

From figure 4, we can note that to find the optimal solution, the generation number increases drastically as the number of cities increases. But the shortest tour found from these results is almost as same as the optimal solution. So we can conclude that this proposed algorithm can perform better for smaller data inputs. Its performance degrades when the number of cities increases over time.

For the experimental result, we used a known TSP problem named 'berlin52' and two other random data set. 'berlin52' contains 52 coordinates of various locations inside Berlin city as shown in figure 1, 2 and 3.

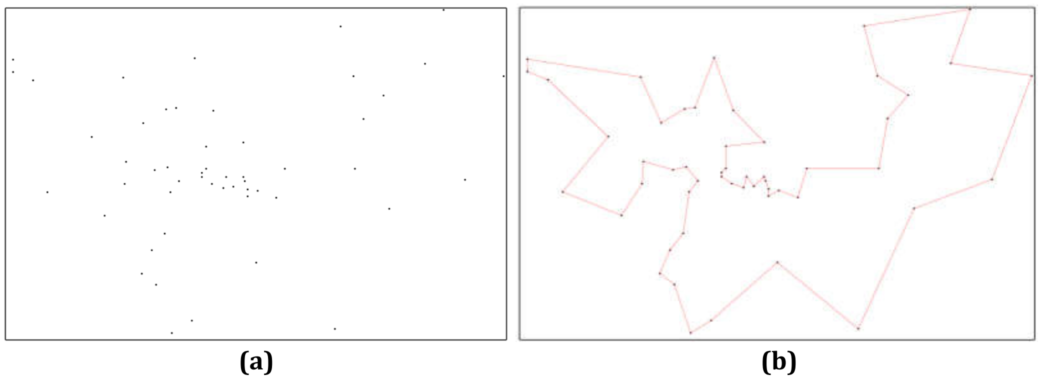


Figure 5.1. Processing Example of (a) Experiment 1
(b) The optimal solution of Experiment 1

The optimal solution to 'berlin52' problem is the best distance to travel to all locations. In our test, the best distance was 7526m. This optimal solution was obtained in 337 generations with 1015 times of mutation.

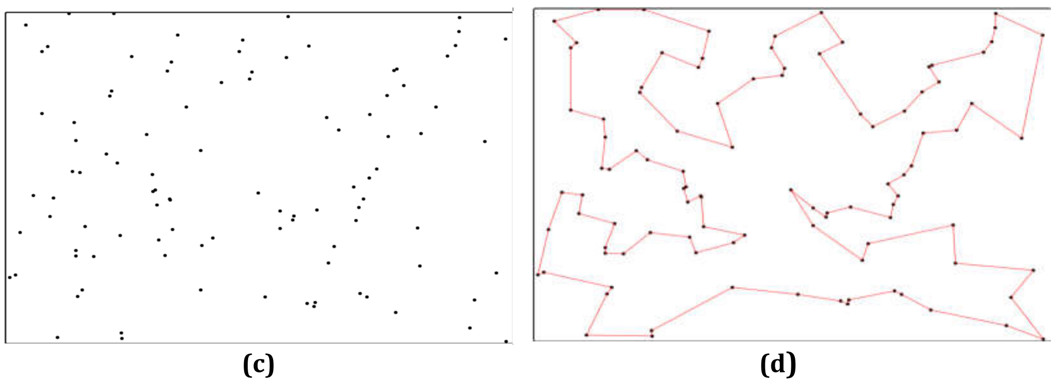


Figure 5.2. Processing Example of (c) Experiment 2
(d) The optimal solution of Experiment 2

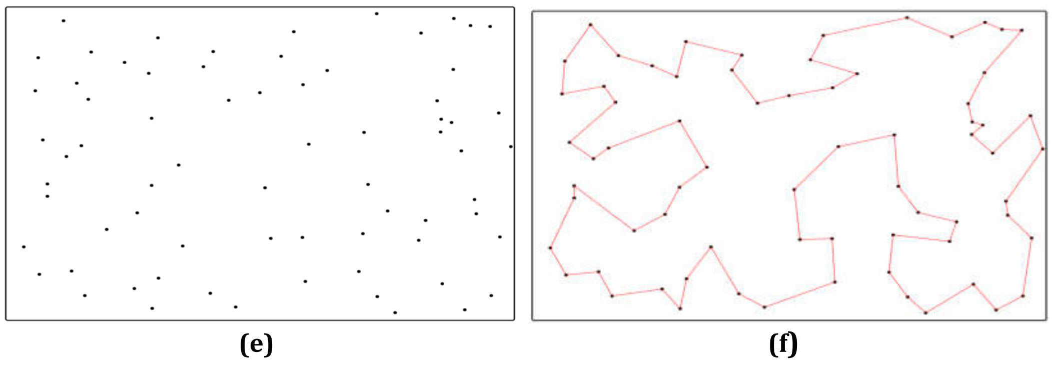


Figure 5.3. Processing Example of (e) Experiment 3
(f) The optimal solution of Experiment 3

To compare the proposed algorithm with other genetic algorithm methods, they are tested one by one on three different data set.

Table 1. Comparison between Proposed GA and other GA methods

Algorithms	Experiment	Cities	Average Generation	Average Mutation	Best result
Proposed GA	Experiment 1 (berlin52)	52	337	1015	7526
Classical GA		52	8654	16822	7537
GA using Ordered Crossover		52	5211	9361	7530
GA using Modified Cycle Crossover		52	1087	3023	7527
Proposed GA	Experiment 2	70	840	2578	5221
Classical GA		70	19279	50137	5238
GA using Ordered Crossover		70	12825	29346	5229
GA using Modified Cycle Crossover		70	2557	7340	5225
Proposed GA	Experiment 3	100	2758	8243	5854
Classical GA		100	57354	120535	5871
GA using Ordered Crossover		100	45120	77542	5860
GA using Modified Cycle Crossover		100	10653	29252	5854

We can see that our proposed algorithm gives high-performance solutions compared to other genetic algorithm methods to solve the traveling salesman problem for the same data set.

7. CONCLUSION

In this research, a genetic algorithm with a modified mutation method and a modified crossover method using Minimal Weight Variable Order Crossover (MWVOSX) operator is proposed and implemented. They were also discussed in the context of the TSP problem. This research will certainly accelerate the developing genetic algorithm. The proposed method produces us high-performance results. We also proposed a modified selection method using Kruskal's minimum spanning tree. However, the selection method could not be implemented.

The performance of our proposed GA method is compared with three other GA methods. From the acquired results, it is noted that the proposed method generates better results than others. It is hoped that this proposed genetic algorithm method can be potentially used in applications.

Acknowledgment

This research was conducted by the authors from Chittagong University of Engineering & Technology, Chittagong, Bangladesh; The International University of Scholars, Dhaka, Bangladesh; Universiti Malaysia Pahang, (UMP) Malaysia; and Verizon Media, California, USA. The work was partially supported by UMP research grant *RDU170397* through *IBM Center of Excellence* at UMP.

REFERENCES

[1] Ezugwu, A. E. S., & Adewumi, A. O. **Discrete symbiotic organisms search algorithm for travelling salesman problem**. *Expert Systems With Applications*, *87*, 70-78. 2017.

[2] Chudasama, C., Shah, S. M., & Panchal, M. **Comparison of parents selection methods of genetic algorithm for TSP.** In *International Conference on Computer Communication and Networks CSI-COMNET, Proceedings* (pp. 85-87). 2011.

[3] Xu, S., Wang, Y., & Huang, A. **Application of imperialist competitive algorithm on solving the traveling salesman problem**. *Algorithms*, *7*(2), 229-242. 2014.

[4] Srinivasan, K., Satyajit, S., Behera, B. K., & Panigrahi, P. K. **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**. *arXiv preprint arXiv:1805.10928*. 2018.

[5] Hatamlou, A. **Solving travelling salesman problem using black hole algorithm**. *Soft Computing*, *22*(24), 8167-8175. 2018.

[6] Abdoun, O., & Abouchabaka, J. **A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem**. *arXiv preprint arXiv:1203.3097*. 2012.

[7] Hussain, A., Muhammad, Y. S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., & Gani, S. **Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator**. *Computational intelligence and neuroscience*, *2017*.

[8] Dwivedi, V., Chauhan, T., Saxena, S., & Agrawal, P. **Travelling salesman problem using genetic algorithm**. *IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI*

2012), *1*, 25. 2012.

[9] Islam, M. L., Pandhare, D., Makhthedar, A., & Shaikh, N. **A Heuristic Approach for Optimizing Travel Planning Using Genetics Algorithm**. *International Journal of Research in Engineering and Technology eISSN*, 2319-1163. 2014.

[10] Karaboga, D., & Gorkemli, B. **Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms**. *International Journal on Artificial Intelligence Tools*, *28*(01), 1950004. 2019.

[11] Dorigo, M., & Gambardella, L. M. **Ant colony system: a cooperative learning approach to the traveling salesman problem**. *IEEE Transactions on evolutionary computation*, *1*(1), 53-66. 1997.