ESTEEM Academic Journal
Vol. 11, Special Issue 1, March 2015, 127-134

NEW GENETIC OPERATOR FOR SOLVING THE TRAVELLING SALESMAN PROBLEM

Fadzilawani Astifar Alias[1], Maisurah Shamsuddin [2], Siti Asmah Mohamed[3] and Siti Balqis Mahlan[4]

[1,2,3,4]*Department of Computer and Mathematical Sciences, Universiti Teknologi MARA Pulau Pinang, 13500 Permatang Pauh, Pulau Pinang, MALAYSIA.*
[1]*fadzilawani.astifar@ppinang.uitm.edu.my;* [2]*maisurah025@ppinang.uitm.edu.my;*
[3]*sitiasmah109@ppinang.uitm.edu.my;* [4]*sitibalqis026@ppinang.uitm.edu.my*

ABSTRACT

The Travelling Salesman Problem (TSP) is a well-known and important combinatorial optimization problem. The goal is to find the shortest distance tour that visits each city in a given list exactly once and then returns to the starting city. TSP is an NP-complete problem that has many interaction variables with a high degree of freedom. The main objective of TSP is to determine the network route to minimize the total distance, cost or time. In this research, the heuristic method called Genetic Algorithm (GA) is used to solve the TSP. GA is a system developing methods that uses the natural principle of a genetic population and involves three main processes that are crossover, mutation and inversion. GA is implemented with some new operators called Nearest Fragment (NF) and Modified Order Crossover (MOC). GA implementation on TSP is done by using Microsoft C++ Programming. Solutions to the problem are presented and performance comparison is described with the existing best solution.

Keywords: Genetic Algorithm; Travelling Salesman Problem; Distance; Population; Network Route.

1. INTRODUCTION

Genetic Algorithms (GA) were developed initially by Holland and his associates at the University of Michigan in the 1960s and 1970s, and the first full, systematic (and mainly theoretical) treatment was contained in Holland's book Adaptation in Natural and Artificial Systems published in 1975. Goldberg gives an interesting opinion on some of the practical work carried out in this era. Among the early applications of GA were those developed by Bagley for a game-playing program, by Rosenberg in simulating biological processes, and by Cavicchio for solving pattern-recognition problems.

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to the problem with genetic algorithms is evolves. Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offsprings) are chosen

p-ISSN 1675-7939; e-ISSN 2289-4934

according to their fitness (the more suitable they are the more chances they have to reproduce).

This is repeated until some condition (for example, number of populations or improvement of the best solution) is deemed satisfying. The GA are adaptive learning heuristic and they are generally referred to in plural, because several versions that exist are adjustments to different problems. They are also robust and effective algorithms that are computationally simple and easy to implement.

Hopfield and Tank solved the TSP using the neural network. It is important to observe that the network or graph defining the TSP is very different from the neural network itself. As a consequence, the TSP must be mapped, in some way onto the neural network.

1.1 A Basic Genetic Algorithm Implementation

The success of GA is determined by the parameter values such as population size, string length, crossover point and also the choice of operators used. A basic Genetic Algorithm implementation namely the way the initial solution is generated, the fitness function, the GA operators such as crossover, mutation and inversion, with the new operators (NF_MOC), selection mechanism and the stopping criteria used in this research will be discussed.

1.2 The characteristics of Genetic Algorithm

The characteristics of GA that distinguish them from the other heuristics, are follow:

i. GA work with coding of the solutions instead of the solution themselves. Therefore, a good, efficient representation of the solutions in the form of a chromosome is required.

ii. They search from a set of solutions, different from other metaheuristics like Simulated Annealing and Tabu Search that start with a single solution and move to another solution by some transition. Therefore they do a multi directional search in the solution space, reducing the probability of finishing in a local optimum.

iii. They only require objective function values, not continuous searching space or existence of derivatives. Real life examples generally have discontinuous search spaces.

iv. GA are nondeterministic, i.e. they are stochastic in decision, which makes them more robust.

v. They are blind because they do not know when they have found an optimal solution.

1.3 The GA Algorithm

A basic genetic algorithm can be summarized in the following steps:

STEP 1: Initialize a population of chromosomes (a set of solutions).
STEP 2: Evaluate each chromosome in the populations with a suitable function.
STEP 3: Create new chromosomes by mating current chromosomes using suitable operators and also the new operators.

p-ISSN 1675-7939; e-ISSN 2289-4934

STEP 4: Delete some old chromosomes to maintain population size.
STEP 5: Evaluate the new chromosomes and insert them into the population.
STEP 6: If certain stopping criteria are met STOP, otherwise go to step 3.

The flow chart of GA for TSP is shown in Figure 1.

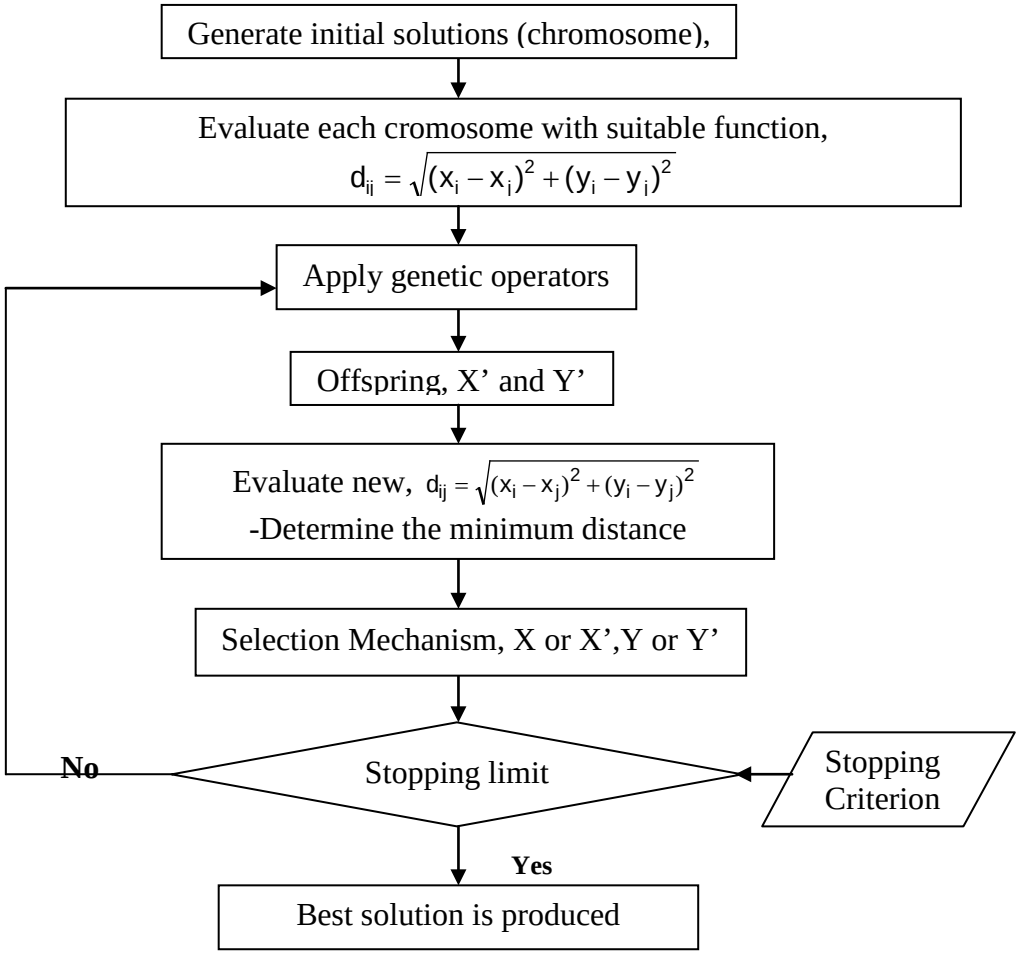

Figure 1: Sample of Outline GA for TSP

1.4 Chromosome Representation

In this research, the selection strategy is the roulette wheel. It uses the fitness value of each chromosome to compute its chance to be selected as a parent of the next generation.

0	**3**	**2**	**5**	**6**	**0**	**4**	**1**	**7**	**0**

Figure 2 : Sample of a GA chromosome

1.5 Fitness Function

p-ISSN 1675-7939; e-ISSN 2289-4934

The fitness reveals the quality of each chromosome. Each individual or solution is assigned to a numerical evaluation of its fitness by a fitness function evaluation. The fitness function used in this research is the objective function. The objective function is calculated by adding the sum of the distances of all nodes. The distance is calculated from the direct distance of each preceding node. We use $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ to generate a distance matrix.

2. NEW GENETIC OPERATORS

We use the new genetic operators called, NF_MOC which are a combination of nearest fragment (NF) and modified over crossover (MOC) to generate a new offspring for the next generation. An ordinary crossover is applied to create the next generation, a two node exchange is used to mutate the selected solution and inversion. To create the first generation, a string with the length of *number of nodes + nearest node which is the first and the last point for every fragment* is created. The total distances between parents and offspring are calculated and the minimum distance will be chosen for the next operation.

2.1 Genetic Algorithm with new operators (NF_MOC)

2.1.1 Crossover

In nature, crossover occurs when two parents exchange parts of their corresponding chromosome. To do that, at least two point crossover needs to be used. In this research, we use two-point crossover. The case of two point crossover is shown in Figure 3.

Parent 1: (7 6 5 | 4 3 2 | 1) ⟶ Offspring 1: (7 6 5 5 7 6 1)
Parent 2: (1 3 4 | 5 7 6 | 2) ⟶ Offspring 2: (1 3 4 4 3 2 2)

Figure 3 : Sample of two point crossover

Ignoring the importance crossover:

4 ⟷ 3 ⟷ 7 this gives: 4 ⟷ 7 and 3 ⟷ 3; 2 ⟷ 6

The new generation is:

Offspring 1: (4 2 5 7 3 6 1)

Offspring 2: (1 7 5 3 4 2 6)

2.1.2 Mutation

The mutation procedure uses a two-node exchange where the position of two nodes is exchanged. This mutation procedure is very important since this problem has a lot of local optima. Figure 4 shows an example of a mutation.

p-ISSN 1675-7939; e-ISSN 2289-4934

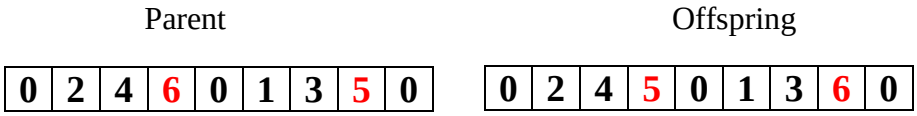

Figure4:Sampleof a mutation

2.1.3 Inversion

We consider a simple inversion in this research. To apply this inversion, we choose *i* and *j*, two random numbers between 1 and the number of nodes, then flip the string placed in the position between *i* and *j*. The inversion operator is described in Figure 5.

Let i=2, j=6

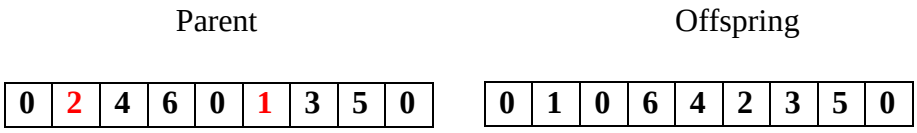

Figure 5 : Sample of an inversion

2.1.4 Illustration of the Procedure

Calculation of total distance

Example:

1 0 16 39 2 44 33 27 11 67 48 61 46 57 0
2 0 21 1 12 34 4 41 38 0
3 0 19 43 22 6 9 18 31 7 20 56 62 68 66 50 55 0
4 0 28 5 40 45 37 47 65 51 49 58 0
5 0 25 17 32 8 10 26 30 42 59 52 54 63 60 0
6 0 14 36 3 23 15 29 24 35 13 53 64 0

i) Total Distance

$$\text{Total Distance} = \sqrt{(x_0 - x_{16})^2 + (y_0 - y_{16})^2} + \sqrt{(x_{16} - x_{39})^2 + (y_{16} - y_{39})^2}$$
$$+\ldots\ldots\ldots + \sqrt{(x_0 - x_{21})^2 + (y_0 - y_{21})^2} + \sqrt{(x_{21} - x_1)^2 + (y_{21} - y_1)^2}$$
$$+\ldots\ldots\ldots\ldots + \sqrt{(x_0 - x_{19})^2 + (y_0 - y_{19})^2}$$
$$+ \sqrt{(x_{19} - x_{43})^2 + (y_{19} - y_{43})^2} + \ldots\ldots\ldots\ldots +$$
$$\sqrt{(x_{53} - x_{64})^2 + (y_{53} - y_{64})^2} + \sqrt{(x_{64} - x_0)^2 + (y_{64} - y_0)^2}$$
$$= 325248$$

p-ISSN 1675-7939; e-ISSN 2289-4934

Below is the illustration of the procedure for the first iteration.We take the data from TSPLIB. TSPLIB introduced by Gerhard Reinelt (1990) is a library that provides us a broad set of test problems and related problems from various sources and types. The results of this implementation are presented in Figure 6.

Iteration: [0]

A: 1 0 16 39 2 44 33 27 11 67 48 61 46 57 0
2 0 21 1 12 34 4 41 38 0
3 0 19 43 22 6 9 18 31 7 20 56 62 68 66 50 55 0
4 0 28 5 40 45 37 47 65 51 49 58 0
5 0 25 17 32 8 10 26 30 42 59 52 54 63 60 0
6 0 14 36 3 23 15 29 24 35 13 53 64 0
Distance of A = 325248

B: 1 0 16 39 27 33 44 2 11 48 67 61 46 57 0
2 0 21 4 1 34 12 41 38 0
3 0 19 43 22 7 31 18 9 6 20 68 66 56 62 50 55 0
4 0 28 5 40 37 45 47 65 51 49 58 0
5 0 17 25 32 8 10 26 30 42 60 63 54 52 59 0
6 0 14 36 24 29 15 13 23 3 35 64 53 0
Distance of B = 316185

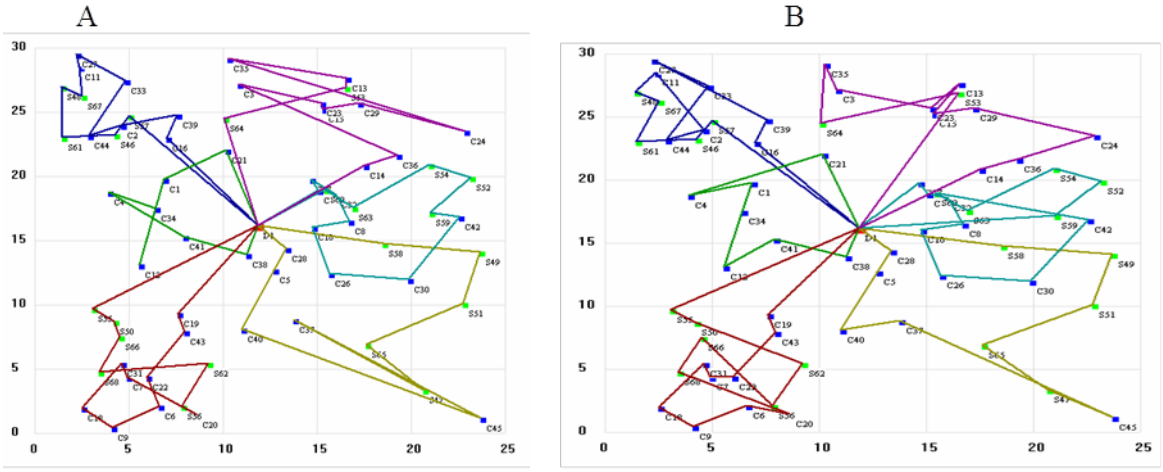

Figure 6 : Sample of Initial Solution A and B

3. RESULTS

Table 1 provides the results obtained by applying GA implementation and the new operators (NF_MOC). We choose 10 sets of data from TSPLIB and the difference in percentage between GA and NF_MOC is detected.

p-ISSN 1675-7939; e-ISSN 2289-4934

Table 1 : Comparison between Genetic Algorithm (GA) and New Operators (NF_MOC)

TSP Sample	Best Solution	Genetic Algorithm (GA)		New Operators (NF_MOC)		Comparison
		Best Solution	%	Best Solution	%	
Lin105	14379	14760	2.650	14754	2.608	0.042
Att48	10628	10752	1.167	10745	1.101	0.066
Bayg29	1610	1695	5.280	1693	5.155	0.124
Berlin52	7542	7621	1.047	7611	0.915	0.133
Bier127	118282	119620	1.131	119620	1.131	0.000
Burma14	3328	3385	1.713	3385	1.713	0.000
Dantzig42	699	726	3.863	712	1.860	2.003
Eil101	629	645	2.544	633	0.636	1.908
Eil51	426	440	3.286	439	3.052	0.235
KroA100	21282	21562	1.316	21562	1.316	0.000
Average			2.400%		1.949%	0.451%

The table shows that the percentage of GA with the new operators (NF_MOC) can obtain the opimal solution without going through many iterations. The difference in percentage between GA and NF_MOC is calculated and we can see that the average percentage is 0.451%.

A personal computer powered by an Intel Core 2 Duo E6400 dual-core processor running at 2.13 Gigahertz is used. The total amount of Random Access Memory installed in the computer is 960 Megabytes. This computer is used when developing the program and also for calculation in order to obtain the results. Table 2 provides the results obtained by applying the new operators of GA implementation with respect to memory space and computing time.

Table 2 : Comparison of GA implementation and GA with the new operators (NF_MOC) with respect to memory space and computing time

TSP Sample	Computing Time (second)		Memory Space (MB)	
	GA	NF_MOC	GA	NF_MOC
Lin105	742	602	7.9 – 8.9	7.0-7.9
Att48	797	777	15.5-17.6	14.4-16.8
Bayg29	616	327	7.9-8.9	6.4-8.2
Berlin52	868	844	5.3-6.0	5.0-6.0
Bier127	860	860	11.2-13.0	11.2-13.0
Burma14	668	668	4.5-5.6	4.5-5.6
Dantzig42	568	520	14.4-16.7	13.8-15.5
Eil101	266	233	7.8-8.9	7.2-8.9
Eil51	648	648	5.6-6.2	5.6-6.2
KroA100	220	198	11.2-13.2	10.5-12.5

p-ISSN 1675-7939; e-ISSN 2289-4934

Table 2 shows that based on the 10 sets of data, if the new operators are used, the optimal value is quickly reached. It means that, the time taken to reach the optimal value is shorter and that is why the NF_MOC uses a small memory compared to the basic GA.

4. CONCLUSION

For basic GA implementation, the initial solutions were generated randomly. In this study, the total distance was calculated. Genetic Algorithms were applied to the population, with three operators which were crossover, mutation and inversion with the new operators called nearest fragment and modified order crossover (NF_MOC). After each operation, the total distance was calculated and the solution with minimum total distance was selected to apply to the next iteration.

After we made the changes to population size, crossover point and operator selection with the new operators, the results were found to be different when we considered two aspects, which were computing time and memory space. We can conclude that, this NF_MOC implementation performs well in terms of distance, time, memory space usage and computing time to find the best solution of optimal value. Therefore, we assume that when the distance and time are minimum, the cost also can be minimized.

REFERENCES

Braun, H. (1990). *On Traveling Salesman Problems by Genetic Algorithms*. 1st Workshop on Parellel Problem Solving From Nature.

Goldberg, David, E. Genetic Algorithms in Search. *Optimization and Machine Learning,* Boston, MA: Kluwer

Hopfield, J. J. (1982). Neural Networks and Physical Systems and Emergent Collective Computational Abilities. Proc. Nat'l Academic of Science : USA.

Lawlor, E. L. *Combinatorial Optimization*. New York

Preds, J. (1993). *Genetic State-Space Search for Constraint Optimization Problems*. Proc. Of the 13th Int. Joint Cont. On Artifial Intelligence (IJCA). San Mateo, USA : Morgan Kaufaman.

Salhi, S. Genetic Algorithms (GA). In Salhi, S. *Topics in Management Mathematics.* The University of Birmingham. 172-185.

Thang, N. B and Byung R. M. (1994). *A New Genetic Approach for the Traveling Salesman Problem*. Proc. IEEE Int. Conf. On Evolutionary Computing : Orlando.

p-ISSN 1675-7939; e-ISSN 2289-4934