Proc. 1st International Conference on
Machine Learning and Data Engineering (iCMLDE2017)
20-22 Nov 2017, Sydney, Australia
ISBN: 978-0-6480147-3-7

Using Genetic Algorithm with Triple Crossover to Solve Travelling Salesman Problem

Shamima Akter[1],Mohammad Sanaullah Chowdhury[2],Subrina Akter[3],Lutfun Nahar[4] and Md. Wahid Murad[5]

[1,2]Department of CSE, University of Chittagong, Bangladesh
[3,4]Department of CSE, International Islamic University Chittagong, Bangladesh
[5]Senior Software Engineer, DataSoft Systems Bangladesh Limited

Corresponding author's E-mail: sana1691@gmail.com

Abstract

TSP is a popular and demanding NP hard problem. Many researchers get interest into it. There are a numerous methods such as SCX, ERX, and GNX etc. for solving TSP with GA. In our paper we proposed a new solution for Traveling Salesman Problem (TSP) using genetic algorithm. A triple crossover technique is applied for finding best & optimal solution of this problem. Triple Crossover Operator (TCO) separate the parent's strings into three substrings comparing cost and derive new partial Offspring containing duplicate nodes then replacing the missing nodes with duplicate nodes allows the system to generate high performance chromosomes. This solution is compared with different well performing Crossover technique. Our Experimental result shows that, due to the well crossover technique has improved performance. Moreover the complexity of this algorithm is negligible.

Keywords: TSP; GA; TCO; SCX; ERX; GNX; substring

1. INTRODUCTION

Travelling Salesman Problem (TSP) is very popular and challenging problem in computer science and computational research. This problem can be stated as, travel all the nodes in a city exactly once and then back to the starting nodes with minimum cost. This is a NP hard problem and cannot be solved exactly in polynomial time. The TSP applied in different situations like automatic drilling of printed circuit boards and threading of scan cells in a testable VLSI circuit as suggested by ravikumar (1992), X-ray crystallography as suggested by zakir (2010), etc.

Many different methods of optimization have been used to solve TSP, like Tabu search, Simulated Annealing, Particle swarm, Hill climbing and Genetic algorithm as suggested by davis (1985) , Po.n and Carter(1995) etc. .Jun Li (1992) proposed an algorithm for solving TSP problem with exclusive and ordinary cities. The algorithm amended two-chromosome format and planned three pairs of crossover and mutation operators. Aybers et al (2009) suggested a method using Euclidean TSP. This is a NP – hard problem associated with resolving the shortest path through a known set of nodes in n-dimensional Euclidean space.
Another existing parallel GA performance to solve the TSP is done by davis (1985). In this paper order crossover and 2-opt mutation method is done. This paper calculates the different elements among populations. The crossover and mutation parts belong to inner parallelization. The differentiation in the crossover part is to create one new Offspring with the combination of two parents. In our paper we choose Genetic Algorithm to solve TSP to find an optimum solution by proposing a new crossover technique.

GAs stores a population of chromosomes, each of which is a candidate solution for its corresponding problem. In each generation (iteration) of the heuristic, several operations are performed on the chromosomes to improve the overall fitness (i.e., cost) of the population.

The genetic algorithm process consists of the following:

1) Encoding: An appropriate encoding is choose for the solution to our problem so that each possible solution has distinct encoding and the some form of a string is called encoding.
2) Evaluation: Next step is to select the initial population, usually at random though different techniques using heuristics have also been proposed. By using the fitness function each individual in the population is then computed. The nearest fittest individual is choosing for the next process.
3) Crossover: The fittest population is used to find the individual's probability of crossover. Crossover is where the two individuals are recombined to create new offspring which are copied into the new population space. In our paper we use a new crossover operation to make new population.
4) Mutation: Next mutation occurs .Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. The character in the corresponding position of the string is changed.
5) Decoding: In this stage the process is complete. Finally a new generation has been formed and the process is repeated until our optimize criteria has been met. At this point the individual which is closest to the optimum is decoded.

2. METHODOLOGY

We Genetic algorithms (GAs) are based essentially on mimicking the survival of the fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology as suggested by goldverg (1989). In order to solve any real life problem by GA, two main requirements are to be satisfied:
(a) A string can represent a solution of the solution space
(b) An objective function and hence a fitness function which measures the goodness of a solution can be defined as suggested by zakir (2010).

2.1. Proposed Genetic Algorithm for Solving TSP

1) At first initial populations of individual strings of nodes are created randomly for the selected TSP problem. Create a cost matrix of the path between every two cities.
2) By using fitness function $F(x) = 1/f(x)$ we assign fittest value to each chromosome.
3) New offspring population is created from two parent chromosomes using proposed crossover operator TCO.
4) Mutation is applied if required.
5) Repeat steps 3 and 4 until we get an optimal solution to the problem.

Below we illustrate our method with an example.

A) Chromosome Design
To solve TSP using GA, at first we need to represent population. In population each chromosome is represented as a sequence of nodes, where each node represent a city and after travelling all nodes once; need to reach the starting node. If the TSP has 6 cities then the chromosome length will be 5.The cost of travelling between two cities is represented as a cost matrix which we use in our paper. Figure 2 shows a sample chromosome for a problem with 6 cities:

B) Fitness Function
The TSP problem is solved by the GA which is used for maximization problem. We need a fitness function that define by, $F(x) = 1/f(x)$

Where, f(x) is the objective function which computes the total cost of a tour represented by a string. Chromosomes are selected with a probability included with their fitness value and then copied into next generation. This process is done in selection process. Highly fittest chromosomes can prevent losing the best found solution.

C) Crossover operator

New solution space is made by creating new offspring's from earlier ones. Then we applied our crossover operator in new solution space. At first, two parents are randomly selected. Then we search for two crossover point, one is from starting and another is from ending side. Then we compare and select the fittest portions from parents. After this, if any duplicate node is found in newly generated chromosome, replace the node with untouched nodes again by comparing the cost with previous adjacent node. The algorithm for this newly crossover technique is as follows:

Step 1: Randomly select two parents P1 and P2 and measure their cost. The representation is illustrates in figure 3.

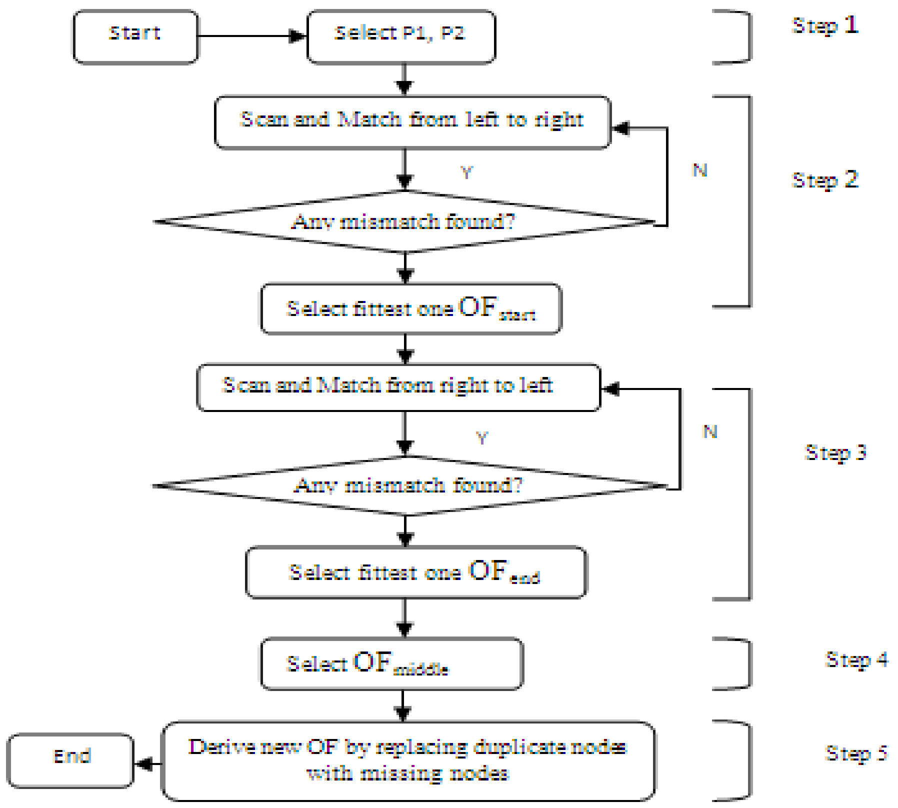
Start
Select P1, P2
Step 1
Scan and Match from left to right
Y
N
Any mismatch found?
Step 2
Select fittest one OF start
Scan and Match from right to left
Y
N
Any mismatch found?
Step 3
Select fittest one OF end
Select OF middle
Step 4
End
Derive new OF by replacing duplicate nodes with missing nodes
Step 5

Figure 1. Flowchart of our proposed method

1 4 3 6 2 5

Figure 2. Chromosome representation

P1	4	3	5	7	6	1	2	Cost=282
P2	4	5	6	1	3	7	2	Cost=329

Figure 3. Parent chromosome representation

Step 2: Start from the first node of both parents and scan from left to right to check whether there is any different sequence or not. If found then stop scanning & compare the cost of last two scanned nodes of both parents and select the fittest one OF (start).
A mismatch is found on 2nd position, so we compare the cost of 4-3(11) and 4-5(59), and select 4-3 as it gives the minimum cost. OF start: 4-3

OF 4 3 ○ ○ ○ ○ ○ Cost=11

Figure 4. Partial chromosome representation of offspring

Step 3: Start from the last node of both parents and repeat step 2 & select the fittest one OF (end).
Now, start from the last node we found an mismatch at 2nd last position so we compare between 1-2(75) and 7-2(31) and select the lowest one.OFend:7-2

OF 4 3 ○ ○ ○ 7 2 Cost=42

Figure 5. Partial chromosome representation of offspring

Step 4: Select OF (middle) by comparing the cost of the parent's no scanned middle substrings.
Now we compare middle nodes which are not scanned 5-7-6(cost-132) from P1 and 6-1-5(cost-71) from P2 and select the minimum one 6-1-5 with 71.
$\text{OF}_{\text{middle}}$: 6-1-5.

OF 4 3 6 1 5 7 2 Cost=230

Figure 6. Final chromosome representation of offspring

Step 5: List out the missing nodes (Mi) from the newly derived immature OF. Replace the duplicate nodes (Di) with the missing nodes (Mi) from left to right by considering the minimum cost between their previous adjacent node (Ai).
If (Ai-Di) < (Ai-Mi)
Select Di
Else
Select Mi
Step 6: Finally Derived new Offspring which requires the lowest cost.
To explicit the method, a problem with 7 cities is considered. Cost matrix is included in table I.

Table 1. Portion of a cost matrix

Node	1	2	3	4	5	6	7
1	100	75	99	9	35	63	8
2	51	100	86	46	88	29	20
3	50	5	100	16	28	35	28
4	20	45	11	100	59	53	49
5	86	63	33	65	100	76	72
6	36	53	89	31	21	100	52
7	58	31	43	67	52	60	100

3. COMPUTATIONAL RESULT AND DISCUSSION

In this age of modern science we concern with time rather than memory. Our proposed method takes a number of calculations but less iteration which utilizes memory but saves time. The previous methods were very good but our proposed method gives optimal solution in less time than the previous methods.

For comparing the efficiency of the different crossover operators, genetic algorithms using SCX, ERX, GNX and TCO have been encoded in Visual C++ on a Pentium 4 personal computer with speed 5 GHz and 1GB RAM under MS Windows 7, and for some TSPLIB instances. Initial population is generated randomly. The following common parameters are selected for the algorithms: population size is 300, probability of crossover is 1.0 (i.e., 100%), probability of mutation is 0.01 (i.e., 1%), and maximum of 15,000 generations as the terminating condition. The experiments were performed 8 times for each instance. The solution quality is measured by the percentage of excess above the optimal solution value reported in TSPLIB website, as given by the formula

$$Excess(\%) = \frac{(Solution\,Value\; - \;Optimal\;\; Solution)}{Optimal\,Solution\,ValueValue} \times 100\% \tag{1}$$

We report percentage of excess of best solution value and average solution value over the optimal solution value of 10 runs. The table also reports the average time of convergence (in second) by the algorithms.

Table 2. Summary of the results of different crossover operators for asymmetric TSPLIB instances

Instance	n	Opt.	ERX	GNX	SCX	TCO
		Sol.	Avg	Avg	Avg	Avg
			Time	Time	Time	Time
br17	17	39	2.10	0.26	0.11	0.9
ftv33	34	1286	70.22	7.64	2.25	1.98
ftv35	36	1473	76.39	1.48	9.69	6.92
ftv38	39	1530	160.87	4.03	6.89	7.12
p43	43	5620	213.99	22.33	22.98	20.22
ftv44	45	1613	157.23	18.46	19.22	15.34
ftv47	48	1776	200.70	42.67	25.99	18.10
ry48p	48	14422	185.59	39.33	25.73	27.31
ft53	53	6905	122.75	29.35	36.73	31.11
ftv55	56	1608	328.59	23.74	35.11	35.21
ftv64	65	1839	326.95	91.39	76.56	81.01
ft70	70	38673	561.14	90.13	74.19	73.99
ftv70	71	1950	432.31	135.97	58.69	58.14
kro124p	100	36230	542.57	178.64	142.02	140.01
ftv170	171	2755	526.46	483.21	259.60	248.91

We also measure the performance of different crossover operators for the instances ftv170 and for some symmetric TSPLIB instances (considering 500 generations). For these instances our proposed method perform better than the previous methods.

4. CONCLUSION AND FUTURE WORK

Solving TSP by Genetic Algorithm is purely depending on the way problem is encoded and which crossover and mutation technique is used. Here we proposed a new crossover operator to solve TSP along with other steps of Genetic Algorithm. In comparison with other crossover operator like SCX, ERX as defined by zakir (2010) our proposed technique shows the best result in terms of times. We use a local search technique to select initial population. In future we want to find a search technique to minimize initial population with good quality of chromosomes which can give a better result by minimizing iteration as well as cost and time.

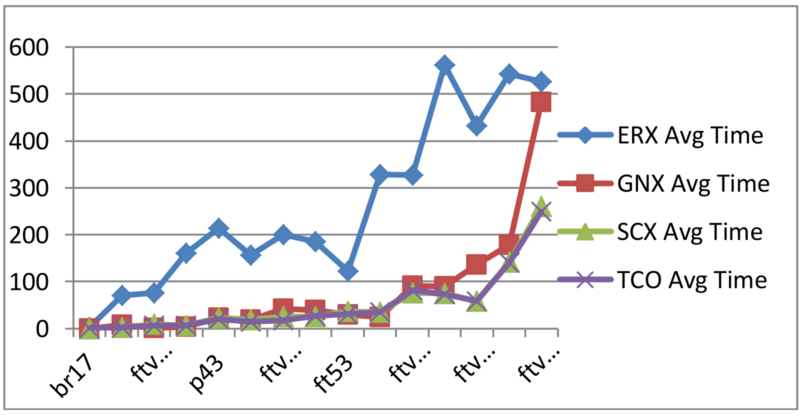

Figure 7. Performance analysis of different crossover operators for asymmetric TSPLIB instances

5. REFERENCES

Jun L, Qirui S, MengChu Z, Xianzhong D (1992). A New Multiple Traveling Salesman Problem and its Genetic Algorithm-based Solution.

C.P. R (1992), Solving Large-scale Travelling Salesperson Problems on Parallel Machines, Microprocessors and Microsystems 16(3), pp. 149-158.

Zakir H. A (2010). Genetic Algorithm for the TSP using Sequential Constructive Crossover Operator, International Journal of Biometrics & Bioinformatics (IJBB), Volume (3), Issue (6).

Aybars U, Serdar K, Ali C, Muhammed C, Ali A (2009). Genetic algorithm based solution for TSP on a sphere, Mathematical and Computational Applications, Vol. 14, No. 3, pp. 219-228.

L. D (1985). Job-shop scheduling with Genetic Algorithms, Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 136-140.

C. M, J. K, G. C and Y. S (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints, European Journal of Operational Research 140, pp. 606-617.

N.J. R and P.D. S (1995). Formae and variance of fitness, In D. Whitley and M. Vose (Eds.) Foundations of Genetic Algorithms 3, Morgan Kaufmann, San Mateo, CA, pp. 51-72.

D.E. G (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, New York.