Comparison Study of Multiple Traveling Salesmen Problem using Genetic Algorithm

Shalini Singh1† and Ejaz Aslam Lodhi2††,

Indira Gandhi Delhi Technical University for Women, New Delhi, India

Summary
Multiple traveling salesman problems (MTSP) are a typical computationally complex combinatorial optimization problem, which is an extension of the famous traveling salesman problem (TSP). The MTSP can be generalized to a wide variety of routing and scheduling problems. The paper makes the attempt to show how Genetic Algorithm (GA) can be applied to the MTSP with ability constraint. In this paper, we compare MTSP in term of distance and iteration by considering several set of cities. The computational results show that the proposed algorithm can find competitive solutions within rational time, especially for large scale problems.

Key words:
Genetic Algorithm, GA Operator, MTSP, TGA, TSP

1. Introduction

From this The TSP is classified as Symmetric Traveling salesman problem (STSP), asymmetric traveling salesman problem (ATSP), and multi traveling salesman problem (MTSP). In this paper we present description about these three widely studied TSP [1].
The STSP is the problem of finding a minimal length closed tour that visits each city once. In this case cities are given by their coordinates (xi , yi) and drs is the Euclidean distance between r and s then we have an Euclidean TSP.
ATSP: If drs ≠ dsr for at least one (r,s) then the TSP becomes an ATSP.
MTSP: The MTSP is defined as: In a given set of nodes, let there are m salesmen located at a single depot node. The remaining nodes (cities) that are to be visited are intermediate nodes. Then, the MTSP consists of finding tours for all m salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized. The cost metric can be defined in terms of distance, time, etc.
MTSP problem is defined on a graph G=(V,A)[2]

- V is the set of n nodes
- A is the set of edges (cities)
- i,j are indices for each cities
- xij is a binary variable
  - xij=1 if arc(i,j) used on the tour
  - xij=0 if arc(i,j) otherwise
- cij distance between city-i and city-j

Objective:
Min ∑∑cijxij

Constraints**:**

- Every salesman must start and end at the same node
- Next two constraints assure that *m* number of salesmen must depart and return back to starting point (node 1):

$$\sum_{j=2} x1j = m$$

$$\sum_{j=2} xj1 = m$$

$$\sum_{i=1} xij = 1,\ j=2...\ n$$

$$\sum_{j=1} xij = 1\ ,i=2,...,n$$

$$xij€0,1,¥(i,j)€A$$

Sub-tour elimination constraints (Secs):

- With the constraint we've so far, sub-tours will be present in our solution which will result in unconnected/disjoint loops.
- To assure impose connectivity, we add SEC constrains.
- *p* denotes the maximum number of nodes that can be visited by a salesman.

$$ui-uj+p*xi\ j\leq p-1 for 2\leq i/j\leq n$$

2. Related Problem

An equivalent formulation in terms of graph theory is: Given a complete weighted graph (where the vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road), find a Hamiltonian cycle with the least weight.
The basic search procedure is as follows [3]:
S1. Select any single node as initial path.
S2. Test the path for admissibility
S3. If the path so far is admissible, list the successors of last node chosen, and extend the path to the first of these. Repeat step 2.

Manuscript received July 5, 2014
Manuscript revised July 20, 2014

S4. If the path so far is inadmissible, delete the last node chosen and choose the next listed successors of preceding node. Repeat step 2.
S5. If all the extension of given node have been shown inadmissible, repeat step 4.
S6. If all the extension from initial node have been shown inadmissible, then no circuit exists
S7. If the successors of last node are the origin, a Hamilton circuit is formed; if all Hamilton circuit are required, then list the circuit found, mark the partial path inadmissible, and repeat step 4.

The test for admissibility in step 2 is made by Deduction rule: Testing the Partial Path, P, for admissibility consists of classifying the edges of the graph into three sets: D= Deleted edge-those edge which cannot be in any Hamilton Path[4] containing the partial path; R= Required Edges-those edges which must be in every Hamilton path containing the partial path; U= Undecided edges-those edges which cannot yet be classified. The edge in D, R, and U are mutually exclusive, D V R V U = E and edges in P are also in R.

Another related problem is the bottleneck traveling salesman problem (bottleneck TSP): Find a Hamiltonian cycle in a weighted graph with the minimal weight of the weightiest edge[5]. The problem is of considerable practical importance, apart from evident transportation and logistics areas. A classic example is in printed circuit manufacturing: scheduling of a route of the drill machine to drill holes in a PCB. In robotic machining or drilling applications, the "cities" are parts to machine or holes (of different sizes) to drill, and the "cost of travel" includes time for retooling the robot (single machine job sequencing problem).

The generalized traveling salesman problem deals with "states" that have (one or more) "cities" and the salesman has to visit exactly one "city" from each "state". Also known as the "traveling politician problem". One application is encountered in ordering a solution to the cutting stock problem in order to minimize knife changes. The cutting-stock problem is often highly degenerate, in that multiple solutions with the same waste are possible. This degeneracy arises because it is possible to move items around, creating new patterns, without affecting the waste.

The sequential ordering problem [6] deals with the problem of visiting a set of cities where precedence relations between the cities exist.

3. Proposed Algorithm

Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution [3]. Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms are inspired by Darwin's theory about evolution. Genetic algorithms are not too hard to program or understand, since they are biological based. Thinking in terms of real-life evolution may help you understand. Here is the general algorithm for a GA [7]:

- Create a Random Initial State

An initial population is created from a random selection of solutions (which are analogous to chromosomes).

- Evaluate Fitness

A value for fitness is assigned to each solution (chromosome) depending on how close it actually is to solving the problem.

- Reproduce (& Children Mutate)

Those chromosomes with a higher fitness value are more likely to reproduce offspring (which can mutate after reproduction). The offspring is a product of the father and mother, whose composition consists of a combination of genes from them (this process is known as "crossing over".

- Next Generation

If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached [8].

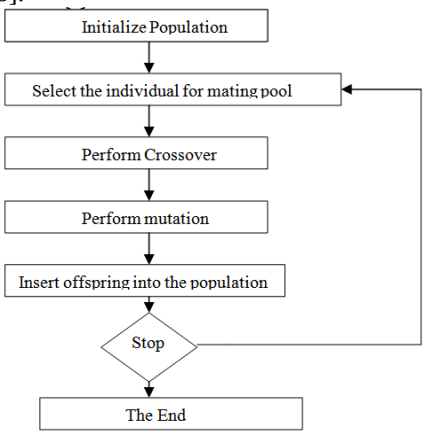

Fig. 1. Genetic Algorithm Work flow

The traveling salesman problem: given a finite number of 'cities' along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point [9].

Genome and Algorithm: The traditional algorithm for the TSP problem, every city must be unique in a gene, and can't be duplicated.

Genes are presented in sequential representation where the cities are listed in the order in which they are visited.

Example: [9 4 3 0 1 2 4 5 6 8 7]

For a crossover operation we selected the Greedy Crossover : Greedy crossover [10] selects the first city of

one parent, compares the cities leaving that city in both parents, and chooses the closer one to extend the tour. If one city has already appeared in the tour, we choose the other city. If both cities have already appeared, we randomly select a non-selected city.

Mutation : Unlike traditional mutation, we swap the order of cities. Example

Before mutation [0 1 2 3 4 5 6]

After mutation [0 1 3 2 4 5 6]

There are a lot of ways of doing such a swapping operation. Easiest way in using random swap. But such a strategy is unable to achieve local optimum. Thus, we used a greedy-swap mutation: The basic idea of greedy-swap[11] is to randomly select two cities from one chromosome and swap them if the new (swapped) tour length is shorter than the old one.

4. Performance Evaluation

Experiments are conducted to evaluate the performance of GA-based Technique. The performance is compared with TSP with Single Salesmen and also the variation in MTSP has been studied. All Algorithms are executed 10 times on each 4 dataset. The experiments focus 2 aspects: quality of solution and computation time. Fig. no. 2,3,4 shows the performance of GA based Multiple traveling salesmen with variable point, closed path and open path giving the city location, distance matrix, best solution history and total distance. Here, we are considering 5 Salesmen with minimum tour is 2. After analyzing all the result of different variations in MTSP, we are now comparing it with TGA.

Variation in TSP with Multiple Salesmen with variable points, closed path and open path using GA

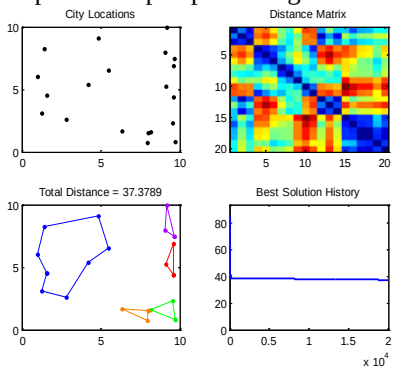

Fig. 2. Result of MTSP using GA

This is the comparison table of MTSP with 4 dataset after taking 10 iterations. Here in Fig. 5 we have studied different MTSP in term of their distance and plot a graph to evaluate their performance. In this Fig. 5, as we increase number of input i.e. cities, variation in different MTSP take place and we observe that MTSP with variable point give optimal result

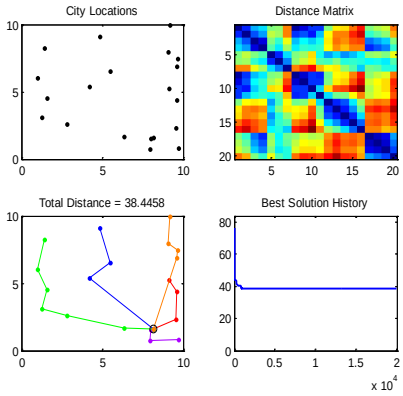

Fig. 3. Result of MTSPFO using GA

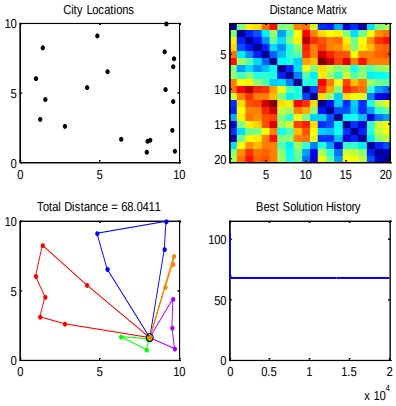

Fig. 4. Result of MTSPFC using GA

Table 1: Comparison of Different MTSP w.r.t Distance

No.of cities	MTSPV Dist.	MTSPFO Dist.	MTSPFC Dist.
20	37.3789	38.4458	67.459
40	56.0354	63.1839	75.5667
60	75.4647	75.9651	91.1436
80	97.9551	83.6589	106.214

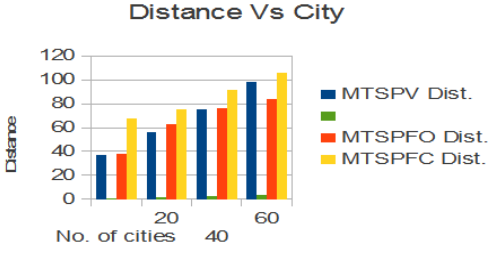

Fig. 5.Comparison graph of Variation in MTSP Distance

Table 2: Comparison of Variation in MTSP w.r.t Iteration

No. of cities	MTSPV Iter.	MTSPFO Iter.	MTSPFC Iter.
20	15709	8699	4106
40	17136	16630	7136
60	17244	9443	17700
80	10239	14266	17329

This is the comparison table of MTSP with 4 dataset after taking 10 iterations. Here in Fig. 6 we have studied different MTSP in term of there iteration with increase in number of cities and plot a graph to evaluate there performance. In both the Fig. 5,6 we have seen that MTSPV justifies our performance evaluation in term of reduce computation time and improved quality against MTSPFO and MTSPFC.

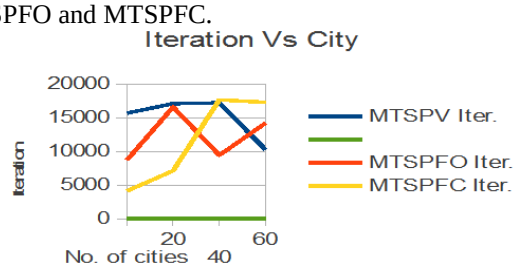

Fig. 6.Comparison graph of Variation in MTSP Iteration

Table 3: Comparison of MTSP_GA and TSP_GA

No. of Cities	MTSP_GA	TSP_GA
20	37.3789	37.4493
40	56.0354	54.1145
60	75.4647	76.1812
80	97.9551	99.1776

This is the comparison table of MTSP with 4 dataset after taking 10 iterations. Here in Fig. 7 we compared Mtsps with TSP in term of there distance and plot a graph to evaluate there performance.

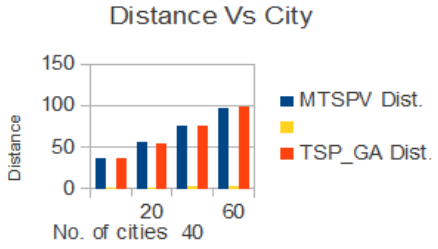

Fig. 7.Comparison graph of TSP_GA and MTSP_GA

5. Conclusion

This paper represents a simple but efficient technique for enhancing the performance of traditional GA. Our simulation result showed that the proposed algorithm outperform TGA in term of not only the computation time but also the accuracy of end result. Evolutionary algorithms like Genetic Algorithms are very successful in solving the TSP. These techniques with some variation can be applied to most of the NP-hard problems in general NP-hard problems can be modeled as graph problems. Moreover, we evaluate the performance of MTSP Variations in term of distance and iteration which shows optimum results and also compare Single Salesmen TSP with Multiple Salesmen TSP. Thus, the result showed that we can significantly reduce the computation time of GA-based algorithm, especially in cases where dataset are large.

References

[1] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. The Traveling Salesman. John Wiley and Sons, 1986.
[2] Bektas,T.(2006).The multiple traveling salesman problem: an overview of formulations and solution procedures. OMEGA: The International Journal of Management Science, 34(3), 209-219.
[3] www.cs.unr.edu
[4] Hamilton Paths in grid graphs by Alon Itai,Christos H. PapaDimitriou and Jayme Luis schwarchifter in 1982 Society for Industrial and Applied Science,Vol.11,no.4,November 1982
[5] Gerard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, 1994
[6] An Inexact Algorithm for the Sequential Ordering Problem, by Escudero in 1988.
[7] www.math.princeton.edu/tsp.
[8] Immune-Genetic Algorithm for Traveling Salesman Problem by Jingui Lu and Min Xie. Nanjing University of Technology,P. R. China.
[9] Gaslab.cs.unr.edu/docs/tech reports/gong/node3.HTML
[10] www.generation5.org/ga.shtml.
[11] Introduction to Genetic Algorithm by Dr. Rajib Kumar Bhattachariya Department of Civil Engineering IIT Guwahati
[12] A Search Procedure from Hamilton path and circuits by Frank Rubin IBM Corporation,Poughkeepsie,New York.

Shalini singh received the B.Tech and M.Tech(Persuing) degrees in Electronics and communication Engineering from M.M.E.C (Mullana) in 2006 and Indira Gandhi Technical University of Women 2013, respectively.