Akshat Agrawal et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 3, Issue 1, March 2016, pp. 50-53

Solution for Traveling Salesman Problem using Genetic Algorithm

Akshat Agrawal

Department of CSE, Amity School of Engineering & Technology, Amity University. Manesar, Haryana

Abstract-The traveling salesman problem (TSP) is to find a tour of a given number of cities (visiting each city exactly once) where the length of this tour is minimized. Testing of every path in an N city tour, would be N!. A 30 city tour would have to measure the total distance of be 2.65 X 1032 different tours. Assuming a trillion additions per second, this would take 252,333,390,232,297 years. Adding one more city would cause the time to increase by a factor of 31. Obviously, this is an impossible solution. Genetic algorithms (GA) are a relatively new optimization technique which can be applied to various problems, including those that are NP-hard. The technique does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time. This, therefore, would be a good algorithm to try on the traveling salesman problem, one of the most famous NP-hard problems.

Keyword: NP Hard, GA, TSP

I. INTRODUCTION

A. The Traveling Salesman problem (TSP)

The traveling salesman problem (TSP) is to find a tour of a given number of cities (visiting each city exactly once) where the length of this tour is minimized. The TSP is defined as a task of finding of the shortest Hamiltonian cycle or path in complete graph of N nodes. It is a classic example of an NP-hard problem. So, the methods of finding an optimal solution

Involve searching in a solution space that grows exponentially with number of city[1].

The traveling salesman problem (TSP) is one of the most widely studied NP-hard combinatorial optimization problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in Operational Research. The simple description of TSP is Give a shortest path that covers all cities along. Let G = (V; E) be a graph where V is a set of vertices and E is a set of edges. Let C = (c_{ij}) be a distance (or cost) matrix associated with E. The TSP requires determination of a minimum distance circuit (Hamiltonian circuit or cycle) passing through each vertex once and only once, it has attracted researchers of various domains to work for its better solutions [3]. Those traditional algorithms such as Cupidity Algorithm, Dynamic Programming Algorithm, are all facing the same obstacle, which is when the problem scale N reaches to a certain degree, the so-called "Combination Explosion" will occur. A lot of algorithms have been proposed to solve TSP. Some of them (based on dynamic programming or branch and bound methods) provide the global optimum solution. Other algorithms are heuristic ones, which are much faster, but they do not guarantee the optimal solutions. The TSP was also approached by various modern heuristic methods, like simulated annealing, evolutionary algorithms and tabu search, even neural networks. In the new algorithm i will use new strategies including selection operator, replace operator and some new control strategy, which have been proved to be very efficient to accelerate the converge speed.

B. Genetic Algorithms

Genetic Algorithms are adaptive search techniques which simulate an evolutionary process like it is seen in nature based on the ideas of selection of the fittest, crossing and mutation. GAs follows the principles of Darwin's theory to find the solution of a problem. The input of a GA is a group of individuals called initial population. The GA following Darwin's theory must evaluate all of them and select the individuals who are better adapted to the environment. The initial population will develop then crossover and mutation.[3].Crossover and mutation are two basic operators of GA. Performance of GA very depend on them. Type and implementation of operators depends on encoding and also on a problem. There are many ways how to do crossover and mutation.

II. LITERATURE SURVEY

A. The Traveling Salesman problem (TSP)

The Traveling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pair wise distances, the task is to find a shortest possible tour that visits each city exactly once.

The Traveling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pair wise distances, the task is to find a shortest possible tour that visits each city exactly once.[2]

B. With metric distances

In the metric TSP, also known as delta-TSP or Δ-TSP, the intercity distances satisfy the triangle inequality. This can be understood as "no shortcuts", in the sense that the direct connection from A to B is never longer than the detour via C:[1]

$C_{ij} <= C_{ik} + C_{kj}$

C. Exact algorithms

The most direct solution would be to try all permutations (ordered combinations) and see which one is cheapest (using brute force search). The running time for this approach lies within a polynomial factor of $O(n!)$, the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. One of the earliest applications of dynamic programming is an algorithm that solves the problem in time $O(n^2 2^n)$.[2]

The dynamic programming solution requires exponential space. Using inclusion–exclusion, the problem can be solved in time within a polynomial factor of 2^n and polynomial space. Improving these time bounds seems to be difficult. For example, it is an open problem if there exists an exact algorithm for TSP that runs in time $O(1.9999^n)$ [2].

D. Genetic Algorithms

Genetic algorithms are an optimization technique based on natural evolution. They include the survival of the fittest idea into a search algorithm which provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result. Genetic algorithms are based on the natural process of evolution. In nature, the fittest individuals are most likely to survive and mate; therefore the next generation should be fitter and healthier because they were bred from healthy parents. This same idea is applied to a problem by first 'guessing' solutions and then combining the fittest solutions to create a new generation of solutions which should be better than the previous generation[7]. I also include a random mutation element to account for the occasional 'mishap' in nature.

E. Outline of the Basic Genetic Algorithm

[Start] Generate random population of *n* chromosomes (suitable solutions for the problem)

[Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population

[New population] Create a new population by repeating following steps until the new population is complete

[Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

[Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

[Accepting] Place new offspring in a new population

[Replace] Use new generated population for a further run of algorithm

[Test] If the end condition is satisfied, **stop**, and return the best solution in current population

[Loop] Go to step [Fitness].[8]

III. PROPOSED SOLUTION

The TSP is defined as a task of finding of the shortest Hamiltonian cycle or path in complete graph of N nodes.

The TSP can be formally described as a graph G = (V,E), where V =(v_1, ..., v_n) is the set of vertices, E = (f(v_i , v_j): v_i, v_j belongs to V)is the set of edges

A. Fitness function

The GAs is used for maximization problem. For the maximization problem the fitness function is same as the objective function. But, for minimization problem, one way of defining a 'fitness function' is as

$$F(x) = 1/f(x)$$

Where *f*(x) is the objective function. Since, TSP is a minimization problem; I Consider this fitness function, where f(x) calculates cost (or value) of the tour represented by a Chromosome.

B. Algorithm

Let we have cities with their coordinates values. Now follow the steps

Step 1: Construct a tree or minimum spanning tree from the graph based on the group size. Root node is the starting point of the salesman.

Step 2: I am going to treat each level of the tree as a chromosome.

Step 3: Applying the GA to this tree.

I. Select any chromosome form the tree
II. Here we have two kind of crossover one is inside the chromosome (tree level) and other in between two chromosomes(tree level).
    a. One point - part of the first parent is copied and the rest is taken in the same order as in the second parent
    b. Two point - two parts of the first parent are copied and the rest between is taken in the same order as in the second parent
    c. None - no crossover, offspring is exact copy of parents

Mutation can be done to the chromosome(tree level) in the following ways.

    d. Normal random - a few cities are chosen and exchanged
    e. Random, only improving - a few cities are randomly chosen and exchanged only if they improve solution (increase fitness)
    f. Systematic, only improving cities are systematically chosen and exchanged only if they improve solution (increase fitness)
    g. None - no mutation

Step 4: Compute the fitness score *f*(x) using fitness function if new route has better fitness score than previous then replace the route.

Step 5: Repeat step 3 and step 4 until better fitness score is obtained or all cities are visited.
Step 6: return the best result.

IV. Simulation Result

There are 2 parameters to control the operation of the Genetic Algorithm:

- **Neighborhood / Group Size** – It gives the number of node to be considered to find the best tour among the selected node at a time. The best 2 tours are the parents. For group size, a high number will increase the likelihood that the really good tours will be selected as parents, but it will also cause many tours to never be used as parents. A large group size will cause the algorithm to run faster, but it might not find the best solution.
- **Mutation(%)** - The percentage that each child after crossover will undergo mutation when a tour is mutated, one of the cities is randomly moved from one point in the tour to another.

The starting parameter values are:

Parameter	Initial Value
Group Size	**5**
Mutation	**3 %**

In this simulation result, I have changed the parameter like group size and mutation percentage and after analyses I observed that, in Fig. 1 with same group size if I increasing the mutation percentage then it slowly decreasing distance value of the tour and in fig. 2 with same mutation percentage if I increasing the group size then first it slowly decreasing the distance value of the tour but after a certain value of the group size, distance value of the tour will increase.

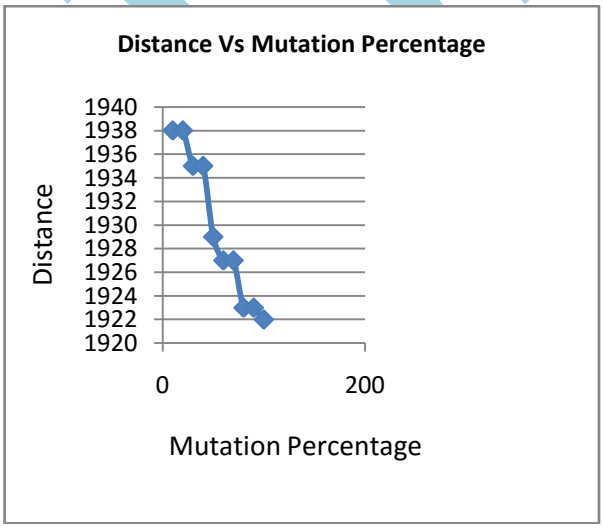

Fig. 1

Sr. no	Mutation Percentage	Total Distance
1	10	1938
2	20	1938
3	30	1935
4	40	1935
5	50	1929
6	60	1927
7	70	1927
8	80	1923
9	90	1923
10	100	1922

Table 1

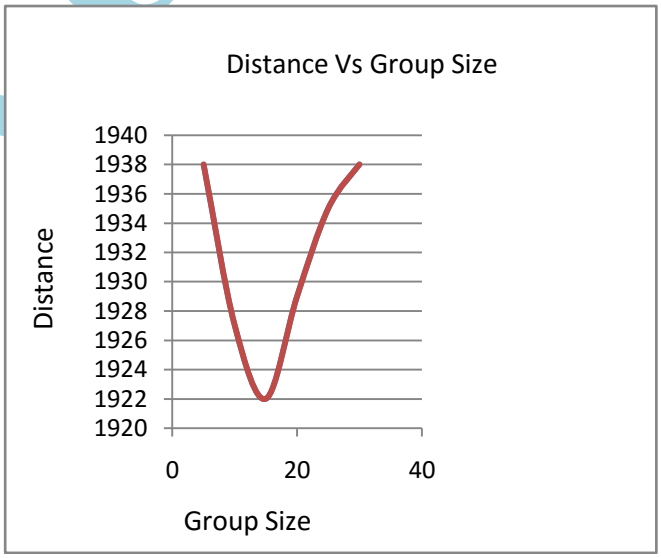

Fig.2

Sr. no	Group Size	Total Distance
1	5	1938
2	10	1927
3	15	1922
4	20	1929
5	25	1935
6	30	1938

Table 2

V. CONCLUSION

In this paper "Solving traveling salesman problem algorithm" based on NP-Hard problem. I have analyzed the result on different parameter like group size and mutation percentage. I observed that with same group size if I increasing the mutation percentage then it slowly decreasing distance value of the tour and with same mutation percentage if I increasing the group size then first it slowly decreasing the distance value of the tour but after a certain value of the group size, distance value of the tour will increase with the group size. A large group size will cause the algorithm to run faster, but it might not find the best solution. In future work I am planning to reduce distance of the tour with further improvement in the algorithm.

REFERENCES

[1]. Introduction to Algorithms,2nd Ed. pp.1027-1033,2001.

[2]. C.H. Papadimitriou and K. Steglitz."Combinatorial Optimization: Algorithms And Complexity".Prentice Hall of India Private Limited, India, 1997.

[3]. X. P. Wang and L. M. Cao, Genetic Algorithm-Theory, Application and Software Realization. Xi'an, Shanxi: Xi'an Jiao Tong University Press,2002.

[4]. Zhu Qiang," A New Co-evolutionary Genetic Algorithm for Traveling Salesman Problem",IEEE ,2008.

[5]. S. Chatterjee, C. Carrera, and L. A. Lynch, "Genetic Algorithms and Traveling Salesman Problems," European Journal of Operational Research, vol. 93,1996.

[6]. Y. Nagata and S. Kobayashi, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in Proc. 7th Int. Conf. Genetic Algorithms (ICGA), 1997, pp. 450–457.

[7]. Budinich, M. (1996), A self-organizing neural network for the traveling salesman problem that is competitive.

[8]. Traveling salesman problem, *Computers & Operations Research*, 30(5): 773-786 Zhu Qiang," A New Co-evolutionary Genetic Algorithm for Traveling Salesman Problem",IEEE ,2008.

[9]. Y. Nagata and S. Kobayashi, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in Proc. 7th Int. Conf. Genetic Algorithms (ICGA), 1997, pp. 450–457.

[10]. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. New York: Addison-Wesley Publishing Company, Inc., 1989.

