International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-3, Issue-2, May 2013

Study of Variation in TSP using Genetic Algorithm and Its Operator Comparison

Shalini Singh, Ejaz Aslam Lodhi

Abstract*— The Purpose of this Paper is to give near optimal solution in terms of quality and computation time. By implementing Genetic Optimization Technique, the effectiveness of the path has been evaluated in terms of fitness function with the parameter such as tour length. In this research work, we see different variation in traveling salesmen problem using Genetic Algorithm Technique. Considering the Limitation of Nearest Neighbor we find that the number of iteration and resulting time complexity can be minimized by using Genetic approach. We also compare the operator of pursued approach which give the best result for finding the shortest path in a shortest time for moving toward the goal. Thus the optimal distance with the tour length is obtained in a more effective way.

Index Terms*—TSP, Fitness Function, Genetic Algorithm, Nearest Neighbour, GA operators.*

I. INTRODUCTION

A TSP is a 'NP-hard' problem first formulated as a mathematical problem in 1930,has been receiving continuous and growing attention in artificial intelligence, computational mathematics and optimization in recent years. TSP can be described as follows: Given a set of cities, and known distances between each pair of cities, the salesman has to find a shortest possible tour that visits each city exactly once and that minimizes the total distance travelled.
The mathematical model of TSP is described below:
Given a set of cities C = {C1, C2, C3… Cn}, the distance of each pair of cities is d(Ci,Cj).The problem is to find a route [C1,C2,C3…Cn] that visits each city exactly once and makes f(x)=minΣdijxij to have a minimum value. Where, f(x) is a fitness function i.e it evaluates each chromosome and sets the numeric value to it, which represents the quality of the chromosome – e.g. of the solution, which the chromosome represents.
Fitness function is then used for evaluating the population and preferring the higher quality of individuals for mating and creating offspring. For TSP, the fitness function of chromosome is computed at the total distance of the represented solution. But even here, computing of the cost of solution is not so easy and could be research, as the total distance equation does not have to be the best fitness function. As TSP algorithm tends sometime to create quite long distance connections, root mean square (RMS) value could be used to compute the cost. RMS value is just sum of square roots of distances between the cities in path which is encoded in chromosome. In this way, we could prefer more expecting solution of (in distances) 2 3 2 2 (total distance=9, RMS cost 21) against chromosome 1 2 1 5 (total distance 9, but RMS distance 31).

Manuscript received on May, 2013.
Shalini Singh, Department of Electronics and Engineering, Indira Gandhi Institute of Technology,Guru Gobind Singh Indraprastha University,New Delhi,India.
Ejaz Aslam Lodhi,,Department of Electronics and Engineering, Indira Gandhi Institute of Technology,Guru Gobind Singh Indraprastha University,New Delhi,India.

For more difficult applications, the fitness function could be defined in complex abstract and non exact way that only tries to compare the quality of chromosomes against each other, but fitness function by itself does not return any meaningful information.
TSP is the problem of the permutation of n cities. For n cities, there should be n! Different permutations. For the symmetric TSP, each route has two different ways to represent. Therefore, the size of its search space is: S =n!/2n= (n-1)!/2. In this paper,we are trying to solve the problem with various algorithms.i.e. Genetic Algorithm (GA), with the advantages of robustness, flexibility and versatility, has been widely studied to solve large-scale combinatorial and optimization problems[6][7].
Genetic algorithms[8] emulate the mechanics of natural selection by a process of randomized data exchange. They were inspired by the behavior of natural systems, the terminology used to describe them is a mix from both biological and computer fields. A genetic algorithm manipulates strings of information, usually called chromosomes. These encode potential solutions to a given problem. Chromosomes are evaluated and assigned a score (fitness value) in terms of how well they solve the given problem according to criteria defined by the programmer. These fitness values are used as a probability of survival during a round of reproduction. New chromosomes are produced by combining two (or more) parent chromosomes. This process is designed to lead to a succession of fitter offspring, each encoding better solutions, until an acceptably good solution is found[4].

II. RELATED WORK

Greedy Algorithms[5] are a method of finding a feasible solution to the traveling salesman problem. The algorithm creates a list of all edges in the graph and then orders them from smallest cost to largest cost. It then chooses the edges with smallest cost first, providing they do not create a cycle. The greedy algorithm gives feasible solutions however they are not always good.
The Nearest Neighbor algorithm[5] is similar to the greedy algorithm in its simple approach. We arbitrarily choose a starting city and then travel to the city closest to it that does not create cycle. We continue to do this until all cities are in is, the edge en1 where n is the number of cities) can be quite large. The nearest neighbor algorithm was one of the first algorithm used to determine a solution to the travelling salesmen problem. In it, the salesman starts at a random city and repeatedly visits the nearest city until all have been visited. It quickly yields a short tour, but usually not the optimal one.
These are the steps of the algorithm:

1) Stand on an arbitrary vertex as current vertex.
2) Find out the lightest edge connecting current vertex and an unvisited vertex V.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

3) Set current vertex to V.
4) Mark V as visited.
5) If all the vertices in domain are visited, then terminate.
6) Go to step 2.

The sequence of the visited vertices is the output of the algorithm The nearest neighbor algorithm is easy to implement and executes quickly, but it can sometimes miss shorter routes which are easily noticed with human insight, due to its "greedy" nature. As a general guide, if the last few stages of the tour are comparable in length to the first stages, then the tour is reasonable, if they are much greater, then it is likely that there are much better tours. Another check is to use an algorithm such as the lower bound algorithm to estimate if this tour is good enough.

In the worst case, the algorithm results in a tour that is much longer than the optimal tour. To be precise, for every constant r there is an instance of the traveling salesman problem such that the length of the tour length computed by the nearest neighbor algorithm is greater than r times the length of the optimal tour. Moreover, for each number of cities there is an assignment of distances between the cities for which the nearest neighbor heuristic produces the unique worst possible tour.

III. PROPOSED ALGORITHM

A. The Concept

GAs simulate the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution[1].GAs are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following foundations:

[1] Individuals in a population compete for resources and mates.
[2] Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
[3] Genes from `good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
[4] Thus each successive generation will become more suited to their environment.

Search Space

A population of individuals are is maintained with in search space for a GA, each representing a possible solution to a given problem. Each individual is coded as a finite length vector of components, or variables, in terms of some alphabet, usually the binary alphabet {0,1}. To continue the genetic analogy these individuals are likened to chromosomes and the variables are analogous to genes. Thus a chromosome (solution) is composed of several genes (variables).A fitness score is assigned to each solution representing the abilities of an individual to `compete'. The individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective `breeding' of the solutions to produce `offspring' better than the parents by combining information from the chromosomes.

The GA maintains a population of n chromosomes (solutions) with associated fitness values. Parents are selected to mate, on the basis of their fitness, producing offspring via a reproductive plan. Consequently highly fit solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. As parents mate and produce offspring, room must be made for the new arrivals since the population is kept at a static size. Individuals in the population die and are replaced by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. In this way it is hoped that over successive generations better solutions will thrive while the least fit solutions die out.

New generations of solutions are produced containing, on average, more good genes than a typical solution in a previous generation. Each successive generation will contain more good `partial solutions' than previous generations. Eventually, once the population has converged and is not producing offspring noticeably different from those in previous generations, the algorithm itself is said to have converged to a set of solutions to the problem at hand.

Based on Natural Selection:

After an initial population is randomly generated, the algorithm evolves the through three operators:

1. selection which equates to survival of the fittest;
2. crossover which represents mating between individuals;
3. mutation which introduces random modifications.

1. Selection Operator[2]

- key idea: give preference to better individuals, allowing them to pass on their genes to the next generation.
- The goodness of each individual depends on its fitness.
- Fitness may be determined by an objective function or by a subjective judgment.

2. Crossover Operator[3]

- Prime distinguished factor of GA from other optimization techniques
- Two individuals are chosen from the population using the selection operator
- A crossover site along the bit strings is randomly chosen
- The values of the two strings are exchanged up to this point
- If S1=000000 and s2=111111 and the crossover point is 2 then S1'=110000 and s2'=001111

S1	0 0	0 0 0 0
S2	1 1	1 1 1 1

S1'	1 1	0 0 0 0
S2'	0 0	1 1 1 1

Fig. 1. Single point Crossover

- The two new offspring created from this mating are put into the next generation of the population
- By recombining portions of good individuals, this process is likely to create even better individuals

3. Mutation Operator

- With some low probability, a portion of the new

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

individuals will have some of their bits flipped.

- Its purpose is to maintain diversity within the population and inhibit premature convergence.
- Mutation alone induces a random walk through the search space
- Mutation and selection (without crossover) create a parallel, noise-tolerant, hill-climbing algorithms

Before	0 1 1 1 1 1 1 0
After	1 1 1 1 1 1 1 0

Fig. 2. Mutation operation

Effects of Genetic Operators

- Using selection alone will tend to fill the population with copies of the best individual from the population
- Using selection and crossover operators will tend to cause the algorithms to converge on a good but sub-optimal solution
- Using mutation alone induces a random walk through the search space.
- Using selection and mutation creates a parallel, noise-tolerant, hill climbing algorithm

B. The Algorithm GA

1. randomly initialize population(t)
2. determine fitness of population(t)
3. repeat
    1. select parents from population(t)
    2. perform crossover on parents creating population(t+1)
    3. perform mutation of population(t+1)
    4. determine fitness of population(t+1)
4. until best individual is good enough

IV. EXPERIMENT RESULTS

Experiments are conducted to evaluate the performance of GA. The performance is compared with TSP using NN Algorithm. To know the performance of each operators Flip, Swap and Slide also are tried to compare. All Algorithm are executed 10 times on each 5 dataset. The experiment focus 2 aspects: quality of solution and computation time. Fig. 3. shows the performance of single traveling salesmen with closed path, giving the city location, distance matrix, best solution history and total distance. .

A. Variation in TSP with Single Salesmen closed loop using GA

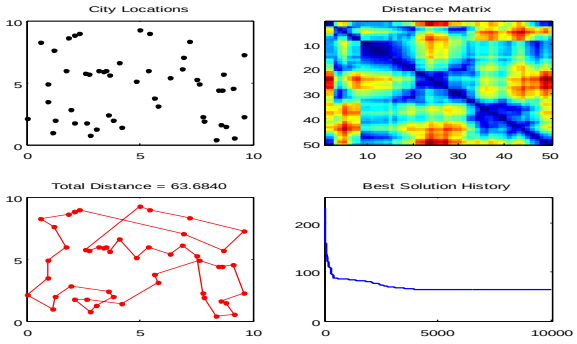

Fig. 3. Result of TSP using GA

B. Variation in TSP with Single Salesmen open loop using GA

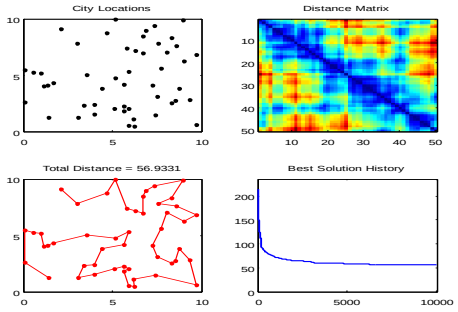

Fig. 4. Result of TSPO using GA

Fig. 4. shows the performance result of single traveling salesmen problem with open path giving city location, Distance matrix, Best solution result and Total distance

C. Variation in TSP using Nearest Neighbor Algorithm

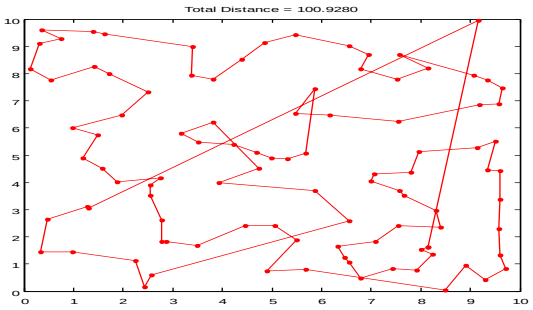

Fig. 5. Result of TSP using NN

Fig. 5. shows the performance result of Nearest Neighbor giving Total distance with 100 population size

D. Comparison Table of TSP_GA and TSP_NN

No. of cities	TSP_GA	TSP_NN
20	37.4493	49.1623
40	50.4738	63.9431
60	64.9016	76.2757
70	68.7479	79.8404
90	79.4125	97.4385

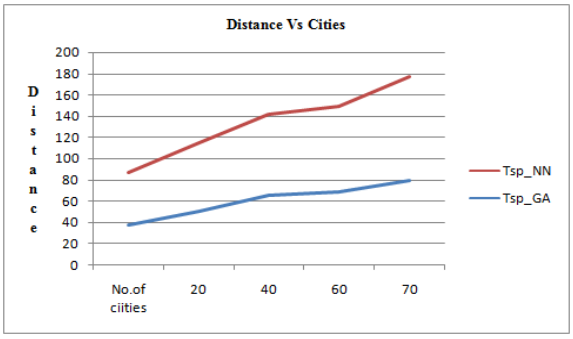

Fig. 6.Comparison graph of TSP_GA and TSP_NN

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

E. TSP_GA(Operator Comparison)

No. of cities	Iteration	Pop_size	GA(Flip)Dist.	GA(Swap) Dist.	GA(Slide) Dist.
30	6000	60	43.3758	50.8658	49.7778
50	10000	100	56.9020	68.1469	66.1501
70	14000	140	70.6605	84.1625	78.3081
90	18000	180	76.8693	112.0468	99.5990
100	20000	200	87.2720	127.9681	112.3047

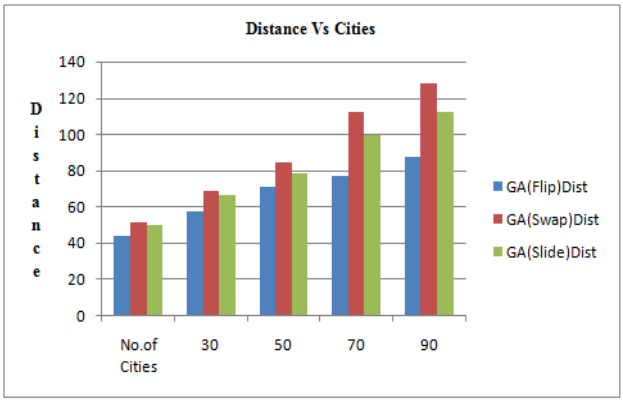

Fig. 7. Comparison graph of different GA Operator

V. CONCLUSION

Genetic algorithms appear to find good solutions for the traveling salesman problem in achieving the goal of decreasing computing time by studying different variations in it. In this paper, we study Traveling Salesmen problem using GA and NN and we find that TSP_GA is giving better result. Also, GA involves three operator Selection, Crossover and mutation, however it depends very much on the way the problem is encoded and which crossover and mutation methods are used. We are comparing these operators to know the better one in respect of reduce computation time and quality of solution. It seems that the methods that use heuristic information (such as the matrix representation and crossover) perform the best and give good indications for future work in this area.

FUTURE SCOPE

Having compared algorithms used to solve TSP, the following limitations should need to persuade in future work. Compare algorithms applied to solve multiple traveling salesman problem with traveling salesman problem using genetic operator and traveling salesman problem using nearest neighbor algorithm. We are also planning to add timer in my result in order to make comparison more efficient because optimal solution leads to reduce computation time with efficient solution.

REFERENCES

[1] An introduction to Genetic Algorithms. *mit press* edited by Melanie Mitchell
[2] http://www.darwins-theory-of-evolution .com
[3] Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley.
[4] A Genetic Algorithm Tutorial: Darrel Whitley, Computer Science Department, Colorado State University, USA.
[5] Gerard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, 1994.
[6] Potvin, J.V. (1996). Genetic Algorithm for Travelling Salesmen Problem, Annals of operational research , vol.63, pp.339-370.
[7] Wei, J.D & Lee, D.T. (2004). Anew approach to genetic algorithm using Genetic Algorithm with priority encoding,Proc. 2004 IEEE Congress on evolutionay computation portland, pp.1457-1464
[8] http://www.obitko.com/tutorials/genetic-algorithms/encoding.php

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication