ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Comparative Analysis of Various Algorithms Used in Travelling Salesman Problem

Suchetha Vijaykumar[1], Sushma L Shetty[2] , Saniha Maria Menezes[3]

Assistant Professor, Department of IT, AIMIT St Aloysius College, Mangalore, India.[1]

Student, II MSC (ST), AIMIT St Aloysius College, Mangalore, India.[2]

Student, II MSC (ST), AIMIT St Aloysius College, Mangalore, India.[3]

ABSTRACT: The Travelling Salesman Problem is a well known problem which has becomea comparison benchmark test for different computational methods. Its solution is computationally difficult, although the problem is easily expressed. TSP is one of the most famous combinatorial optimization (CO) problems and which has wide application background. The aim of our work is to compare existing algorithms that are already tested on benchmark problems from TSPLIB. The objective is to determine which algorithm holds good for various Travelling Salesman Problems.

I. INTRODUCTION

Travelling salesman problem (TSP) is a well known, popular and extensively studied problem in the field of combinatorial optimization and attracts computer scientists, mathematicians and others. Its statement is deceptively simple, but yet it remains one of the most challenging problems in operational research. It also an optimization problem of finding a shortest closed tour that visits all the given cities. It is known as a classical NP-complete problem, which has extremely large search spaces and is very difficult to solve[3].

We will show different examples about the Travelling Salesman Problem.

Example : A salesman is planning a business trip that takes him to certain cities in which he has customers and then brings him back home to the city in which he started. Between some of the pairs of cities he has to visit, there is direct air service; between others there is not. Can he plan the trip so that he begins and ends in the same city while visiting every other city only once, and pays the lowest price in airfare possible? The key to this is not just finding a solution, but an optimal solution, the one with the lowest airfare.

The TSP is stated as, given a complete graph, G, with a set of vertices, V, a set of edges, E, and a cost, c_{ij}, associated with each edge in E. The value c_{ij} is the cost incurred when traversing from vertex i ∈ V to vertex j ∈ V. Given this information, a solution to the TSP must return the cheapest Hamiltonian cycle of G. A Hamiltonian cycle is a cycle that visits each node in a graph exactly once. This is referred to as a tour in TSP terms.

II. RELATED WORK

III.

Some of the Algorithms for TSP:

1. Greedy 2-opt Algorithm:

Step 1: Let S be the initial value that would be provided by the user and z its objective function value. Set S*=s, z*=z, i=1 and j=i+1=2.

Step 2: Transpose Node i and Node j, i<j. Compare the resulting OFV z with z*. If z □□z* and j<n, set j=j+1 and repeat step 2; Otherwise go to step 3.

Step 3: If z<z*, set S*=S, z*=z, i=1, j=i+1=2 and go to step 2. If z □ z* and j=n, set i=i+1, j=j+1 and repeat step2. Otherwise, output S* as the best solution and terminate the process.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Like the 2-Opt algorithm, greedy 2-opt algorithm also considers pair wise exchanges. Initially, it considers transposing Nodes 1 and 2. If the resulting OFV is less than the previous one, two nodes are immediately transposed. If not, the algorithm will go on to Node 3 and evaluate the exchange, and so on until find the improvement. If Nodes 1 and 2 are transposed, the algorithm will take it as an initial solution and repeat the algorithm until it is impossible to improve the solution any further. Greedy 2-opt algorithm makes the exchange permanent whenever an improvement is found and thus consumes less computational time than 2-Opt algorithm. On the other hand, greedy 2-opt produces slightly worse solutions than 2-Opt algorithm.[1]

2. Genetic Algorithm:

A Genetic Algorithm is one of the oldest and most successful optimization technique based on natural of Evolution. It was originally proposed by John Holland ain the 1960s at the University of Michigan, to study the process of evolution and adaption occurring in nature. This uses survival of the fittest approach for selecting the best (fittest) solution from the available solutions.
As the name implies, this algorithm gets its idea from genetics. Thealgorithm works this way.
Step 0 Obtain the maximum number of individuals in the population P andthe maximum number of generations G from the user, generate P solutions forthe first generation's population randomly, and represent each solution as astring[4]. Set generation counter N_{gen}=1.

Step 1 Determine the fitness of each solution in the current generation'spopulation and record the string that has the best fitness.
Step 2 Generate solutions for the next generation's population as follows:
1. Retain 0.1P of the solutions with the best fitness in the previouspopulation.
2. Generate 0.89P solutions via mating, and
3. Select 0.01P solutions from the previous population randomly andmutate them.
Step 3 Update N_{gen}= N_{gen}+1. If N_{gen} □ G, go to Step 1. Otherwise stop[1].

3. Simulated Annealing Algorithm:

Simulated Annealing (SA) is the oldest probabilistic meta-heuristic algorithm and one of the first algorithms having ability to avoid being trapped in local minima. It was first presented by Kirkpatrick, Gelatt and Vecchi in 1983.

The basic idea of simulated annealing(SA) is from the statistical mechanics and is motivated by an analogy of behaviour of physical systems in the presents of a heat bath. This algorithm obtains better final solutions by gradually going from one solution to the next. The main difference of SA from the 2-opt or 3-opt algorithm is that the local optimization algorithm is often restrict their search for the optimal solution in a downhill direction which mean that the initial solution is changed only if it results in a decrease in the OFV. The temperature refers to the state through which the simulated annealing algorithm passes in its search for a better solution. Starting with an initial temperature, we move to the next temperature only when we have reached a frozen state. When a frozen state is reached, the temperature is reduced by a cooling factor r, and the procedure is repeated until a certain predetermined number of temperature steps have been performed.
Step 0: Set *S* = initial feasible solution
z = corresponding OFV
T = initial temperature = 1 *; r* = cooling factor = 0.9
ITEMP = number of times the temperature *T* is decreased = 0
NLIMIT = maximum number of new solutions to be accepted at each temperature = 10n
NOVER = maximum number of solutions evaluated at each temperature = 100n
Step 1: Repeat step 2 *NOVER* times or until the number of successful new solutions is equal to *NLIMIT*
Step2 : Pick a pair of machines randomly and exchange their positions. If the exchange of the position of the two machines results in the overlapping of some other pairs of cities, appropriately modify the coordinates of the center of

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

the concerned machines to ensure no overlapping. If the resulting solution S* has OFV <= *z*, set S* = S and *z* = corresponding OFV. Otherwise, compute *delta* = difference between z and the OFV of the solution S*. and set S*=S with a probability $e^{-delta/T}$.
Step 3: Set *T* = *rT* and *ITEMP* = *ITEMP* + *1*. If *ITEMP* <= 100, go to step 1; otherwise stop[1]

4. Neural Network Algorithm:

Neural Networks is a field of Artificial Intelligence (AI) where we, by inspiration from the human brain, find data structures and algorithms for learning and classification of data. Many tasks that humans perform naturally fast, such as the recognition of a familiar face, proves to be a very complicated task for a computer when conventional programming methods are used. By applying Neural Network techniques a program can learn by examples, and create an internal structure of rules to classify different inputs, such as recognising images[2].

At First we have to pick a random city in the network at the initial loop. As per node creation process,the number of nodes N on the loop grows subsequently. At every step of processing, a city i is surveyed. A complete iteration will take M (number of total cities that has to be visited) sequential steps, for k=1 to k=M, such that every city would get picked once. A gain parameter is used which decreases between two complete iterations. Several iterations would be needed to processed to reach from high value of gain to low value of gain which would give the final result[5].

Surveying the city k contains the following steps:
Step 1. Find a node j_c such that it is close to city k.
Step 2. Shift node j_c and its neighbours on the ring closer to city k. The distance each node has to be shifted is determined by a function f(G,n) where G is the gain parameter and n is the distance that would be measured along the loop between nodes j and j_c.
(neural network algorithms)n=inf (j- j_c (mod N), j_c -j(mod N))
Each node j has to be moved from present position to a new one.
The function f is defined to be f(G,n)=sqrt(1/2)*exp(-n^2/G^2)
This means:
- when G →□, all nodes will be moved towards city k with the same strength sqrt(1/2).
- when G 0,only node j_c will move closer to city k. By reducing the gain at the end of a complete survey is done by G (1-□)G. □ is the only parameter that needs to get adjusted. It is co-related to the number of complete surveys that is needed to be performed to achieve the result, and hence its quality. The gain is reduced from a high initial G_k which will make sure that large moves for all the nodes at every iteration step to a required low G_f for which the network would be stabilized. The total number of iterative that has to be computed from these two fixed values would depend only on M. A node would be duplicated, if it would be chosen as winner for two different cities in the same survey. The new created node would be put in the ring as the neighbour of the winner, and with the same coordinates in the plane. Both the winner and the created node are inhibited. If chosen by a city, an inhibited node will induce no movement at all in the network for this city presentation. It is then re-enabled on the next presentation. This guarantees that the "twin nodes" will be separated by the moves of their neighbours, before being "caught" by cities. The maximum number of nodes created on the loop experimentally appears to be less than 2M. A node would be deleted, if it will not be chosen as the winner by any city during these three complete surveys. This creation-deletion mechanism has provedimportant to the attainment of near-optimum solutions[1].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

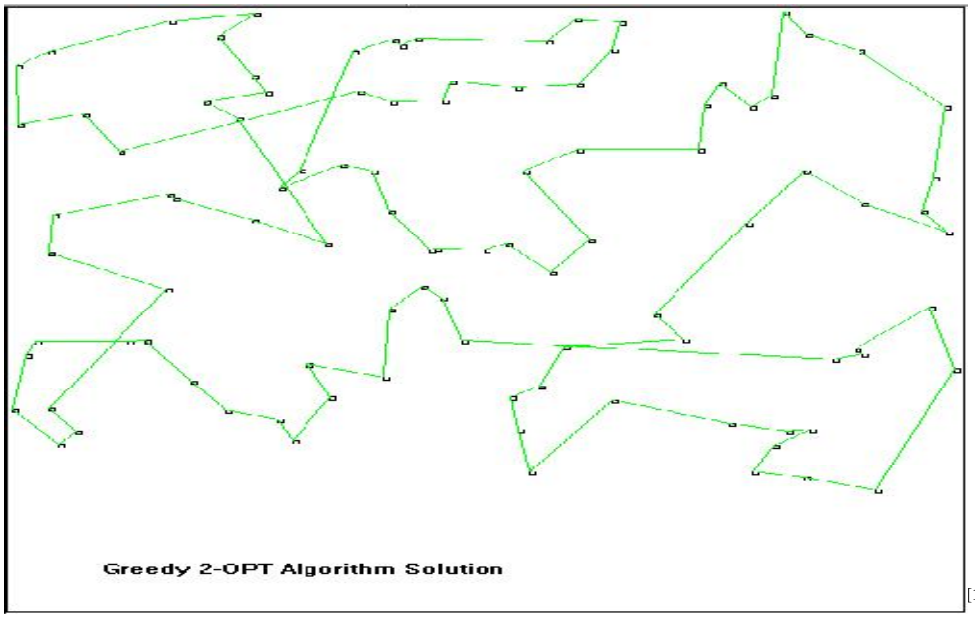
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

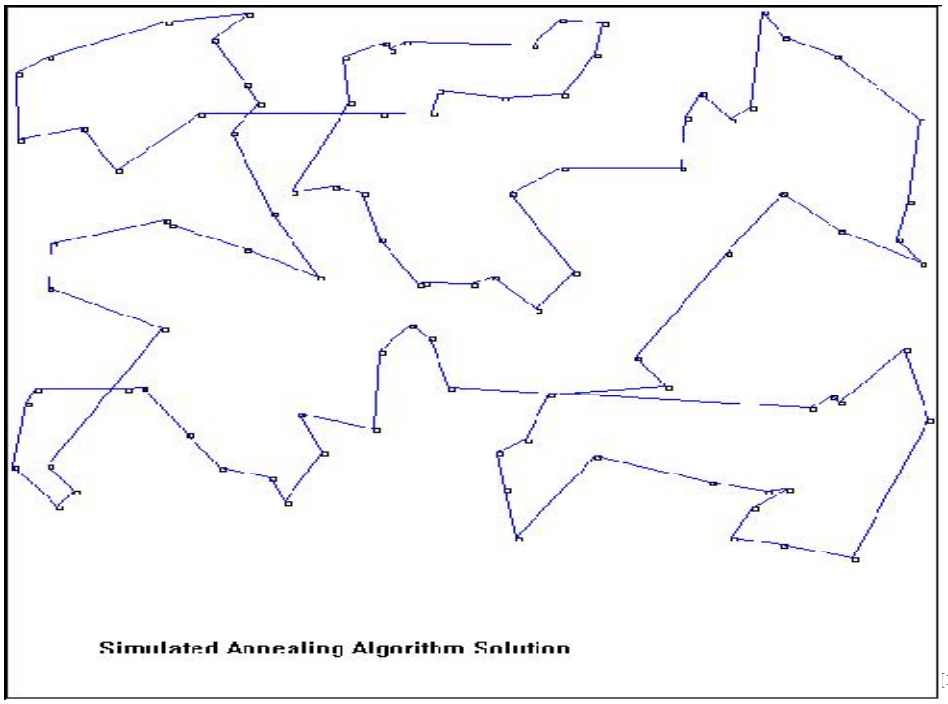
III. RESULT

Greedy 2-opt:



[1]

Simulated Annealing:



[1]

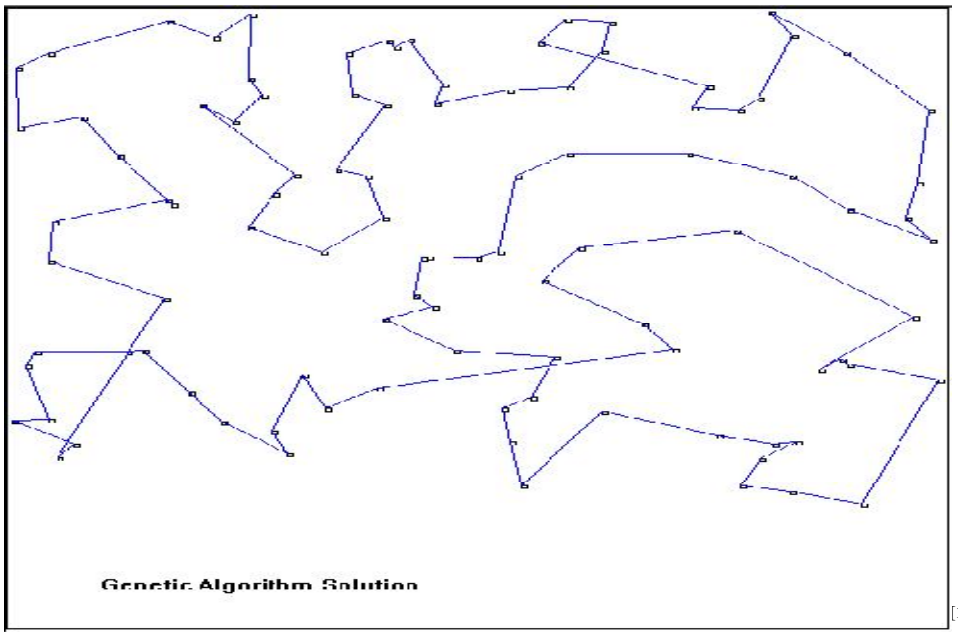
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

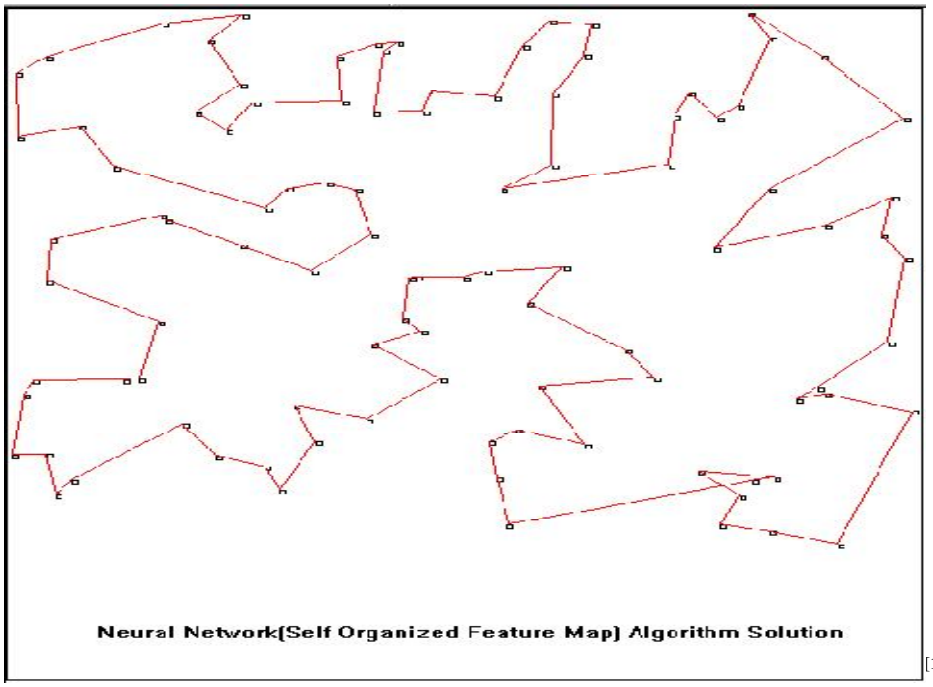
Vol.2, Special Issue 5, October 2014

Genetic Algorithm:



[1]

Neural Network:



[1]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

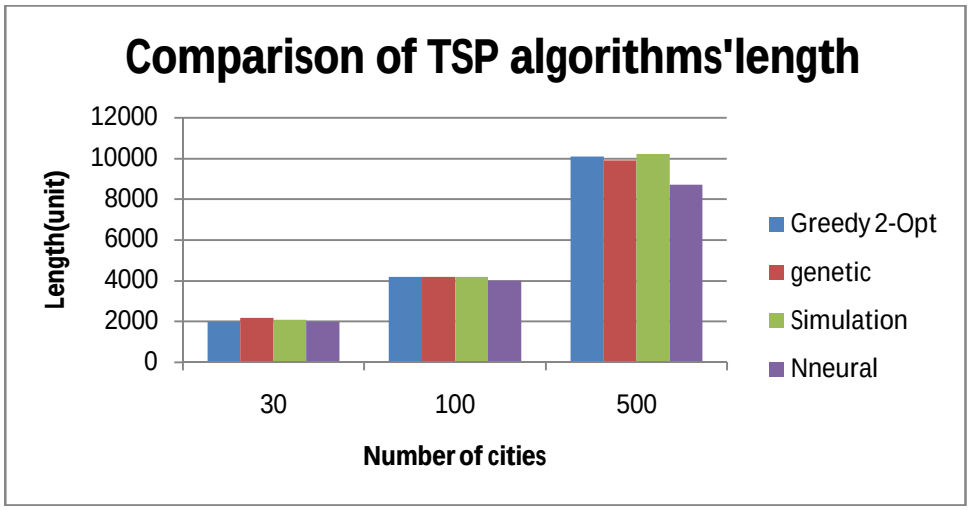
(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

IV. DISCUSSION

We did a performance comparison of 4 algorithms. For Simulation city sizes of 30, 100 and 500 was considered. The following tables and graphs will illustrate the result of simulation. The neural network algorithm came up with the shortest distance for all the number of cities. Even though there are many algorithms such as 3-opt, Greedy 3-opt but due to their longest time consumption, they doesn't give the solution for 500 cities. As the number of cities increases the simulation time also increases for some of the algorithms.

< Length Comparison >



Comparing the above lengths, the length taken by the neural network is less in all the cases i.e. 30, 100 and 500.

< Time Comparison >

Algorithms/City Size	**30**	**100**	**500**
Greedy 2-opt	0.2	0.3	2
Genetic	6	8	34
Simulated Annealing	2	6	27
Neural Network	1	12	346

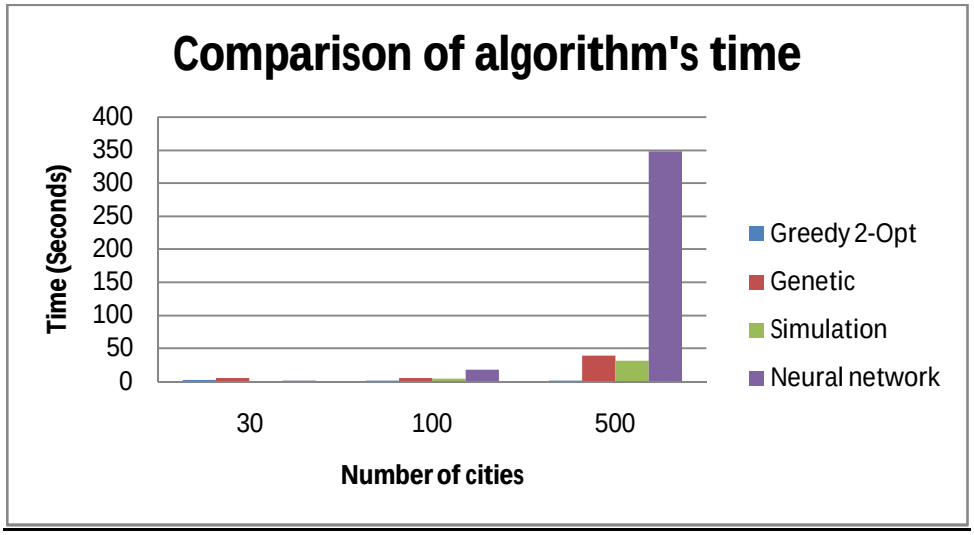
Algorithms/City Size	**30**	**100**	**500**
Greedy 2-opt	2095	4377	10127
Genetic	2295	4302	9906
Simulated Annealing	2177	4320	10279
Neural Network	2086	4035	8778

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014



V.CONCLUSION

The conclusion that is derived from this paper is for smaller size TSP (n<50), greedy 2-opt algorithm will perform well,i.e. high optimality vs. little computational time. The neural network (SOM) algorithm demonstrates the best efficiency for all citysizes. Through the simulation reports observed, the neural network algorithm gives us the shortest distance for 30 cities, 100 cities and 500cities. The number of thecities doesn't affect the optimality of algorithms. Therefore no matter how manycities are involved in the Travelling salesman problem, the Neural networkalgorithm will give the best solution of all these 4 algorithms that werementioned in this paper. For small-size TSP (n<50), improved greedy 2-opt algorithm isrecommended. If the computational time is not a constraint, theneural network is always recommended [6].

REFERENCES

[1]. Kim, B.-I. (n.d.). Comparisons of TSP algorithms.
[2]. neural network algorithms. (n.d.). Retrieved from glyn: http://www.glyn.dk/download/Synopsis.html
[3]. papers. (n.d.). Retrieved from ijarc: http://ijarc.com/papers/Final.docx
[4]. (n.d.). Retrieved from slideshare: http://www.slideshare.net/ahlamansari/analysis-of-algorithm
[5]. (n.d.). Retrieved from ps2netdrivers: http://www.ps2netdrivers.net/manual/thomson.cs84/
[6]. sli7. (n.d.). Retrieved from slideshare: http://www.slideshare.net/ahlamansari/analysis-of-algorithm