Implementation on Genetic Algorithm and Simulated Annealing for solving Travelling Salesman Problem

Shivani Vats[1], Nikita Bakshi[2]
[1,2]*Department of Computer Science Engineering, UIET Lalru, SAS Nagar (India)*

Abstract- **The Travelling Salesman problem is one of the very main problems in Computer Science and Operations Research. It is used to evaluate the lowest cost of doing a work while covering the complete area. In this research article proposed work is a new approach which is simulated annealing and a genetic algorithm for solving Travelling Salesman Problem in order to gather the benefits of SA. Also it diminishes the time that GA spend fixed at local minima. The proposed algorithms tend to produce superior quality results in smallest amount of time.**

Index Terms- Travelling Salesman Problem, Genetic Algorithm, Simulated Annealing, SAGA, ACO.

INTRODUCTION

1. Genetic algorithm for Travelling Salesman Problem: GA begins with arbitrary initiation of a population using chromosomes as abstract presentations of solution candidates. We need to evaluate loop-path which will include each city only once and path length is minimal. The genetic algorithm is series of following operations:

1) Evaluate length of paths
2) Evaluate probabilities
3) Prepare to crossover according probabilities
4) Crossover, parents replaced with children.
5) Mutations

2. Simulated Annealing

There is some well-known classical technique for finding a nearest best solution using linear and non-linear programming for TSP's. SA is based on heuristics from annealing process instead of any classical optimization methods.

Algorithm of simulated annealing

Definitions:

State (s): A particular tour over the set of given cities.

Neighbour State (s'): It is a state access by randomly switching the order of two cities

Cost Function (C): find the total cost of a state

Relative Change in Cost (δ): It is the relative modification in cost c between s and s'

Cooling Constant (β): It is the rate at which the temperature is lowered each time an original solution is found

Acceptance Probability Function (P): determines the probability of moving to a more costly state

n= number of cities or points

T_0= Initial Temperature

T_k= the Temperature at the k^{th} instance of accepting a new solution state.

$T_{k+1} = \beta T_k$, where β is some constant between 0 and 1

$$(\delta, T_k) = \begin{cases} e^{\left(\frac{-\delta}{T_k}\right)} & \delta > 0 \\ 1 & \delta \le 0 \end{cases} \quad for T_k > 0.$$

Note that for $\delta > 0$, for any given T, P is greater for smaller values of δ. In other words, a state s' that is only slightly more costly than s is more likely to be accepted than a state s' which is much more costly than s. Further P decreases with decrease in T

In mathematical terms $\lim_{T\to 0+} e^{\left(\frac{-\delta}{T_k}\right)} = 0$, for $\delta > 0$.

3. Result Analysis:

3.1 Genetic Algorithm for Travelling Salesman Problem:-

3.1.1 Travelling Salesman Problem: A particular salesman travels to each and every cities and completes the route and back to the initial city.

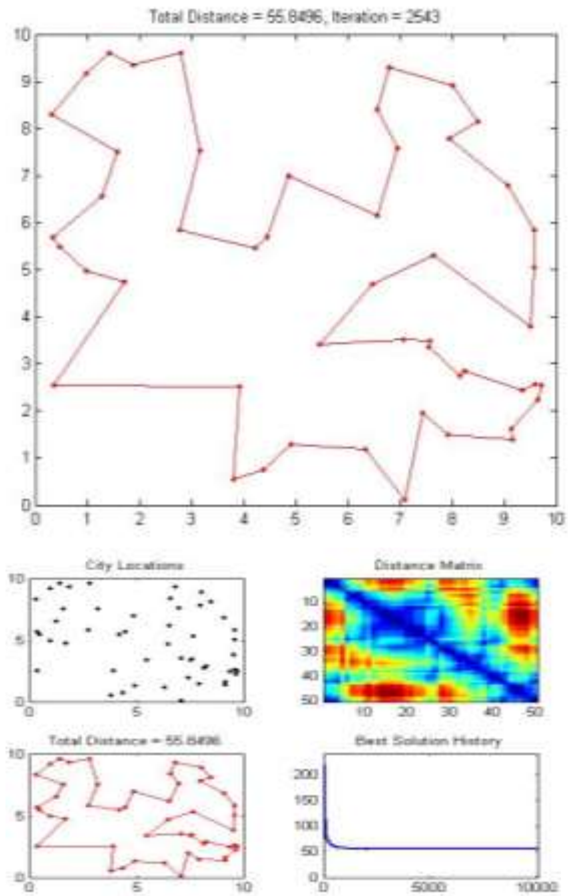
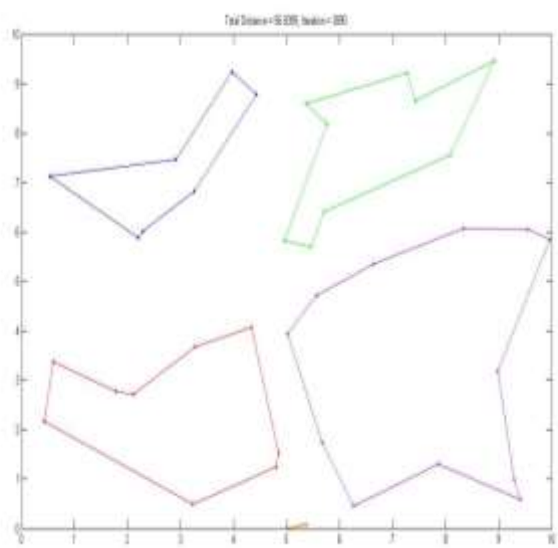


Figure1.1: Best Solution Path for Iteration=2543

3.1.2 Multiple Travelling Salesmen Problem: Every salesman travels to a particular set of cities and completes the route by back to the city from where he started. Each and every city is visited by accurately one salesman.



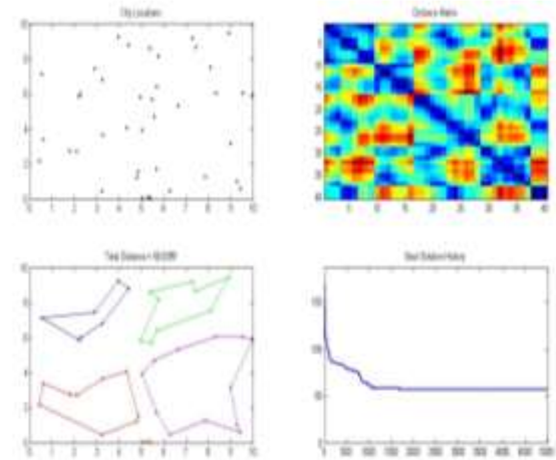
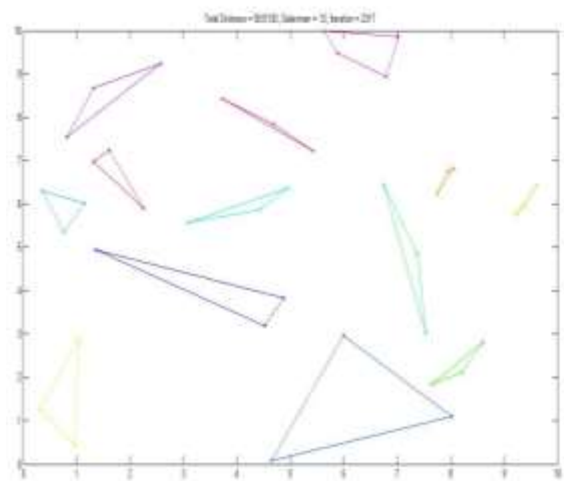


Figure 1.2: Multiple Travelling Salesman Problems for Iteration=3890, Salesman=4



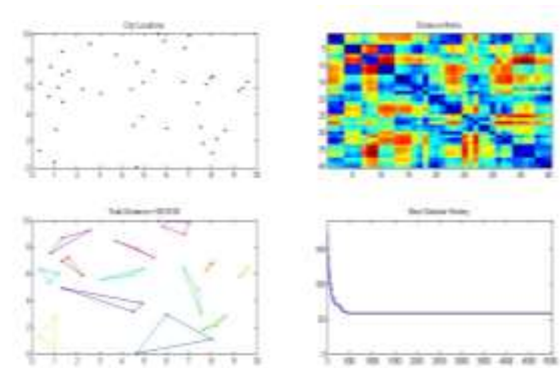


Figure 1.3: Multiple Travelling Salesman Problems for Iteration=2317, Salesman=13

3.1.3. Fixed Start Open Multiple Travelling Salesmen Problem: Every salesman starts at the initial point, but travels to a single set of cities except for the initial point, every city is visited by exactly one salesman.

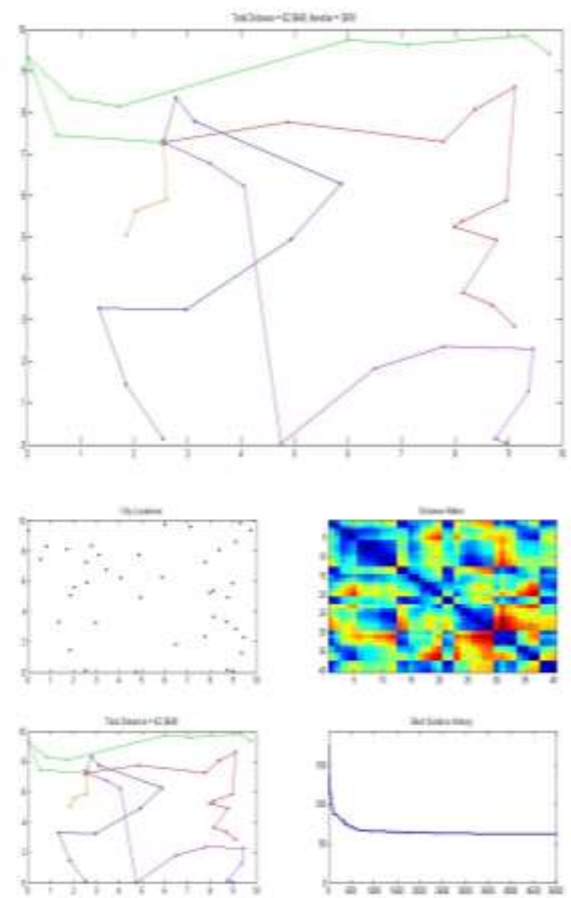
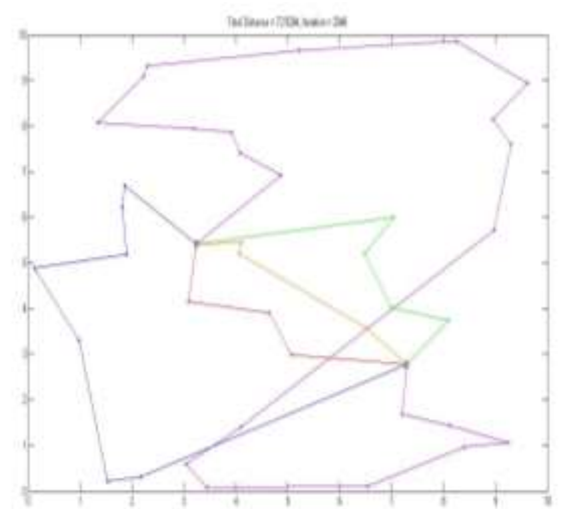


Figure 1.4: Fixed Start Open Multiple Travelling Salesmen Problem for Iteration=3618

3.1.4. Fixed Open Multiple Travelling Salesmen Problem: Every salesman starts at the initial point, and ends at the final point, but travels to a particular set of cities. Except for the initial and final points, each city is visited by exactly one salesman.



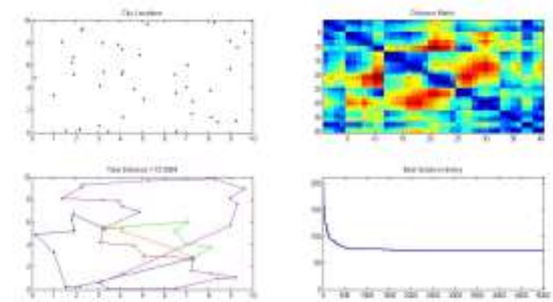


Figure 1.5: Fixed Open Multiple Travelling Salesmen Problem for Iteration=2048

3.1.5. Fixed Multiple Travelling Salesmen Problem: Every salesman starts at the initial point, and terminate at the initial point, but travels to a unique set of cities except for the initial point, every city is visited by exactly one salesman.

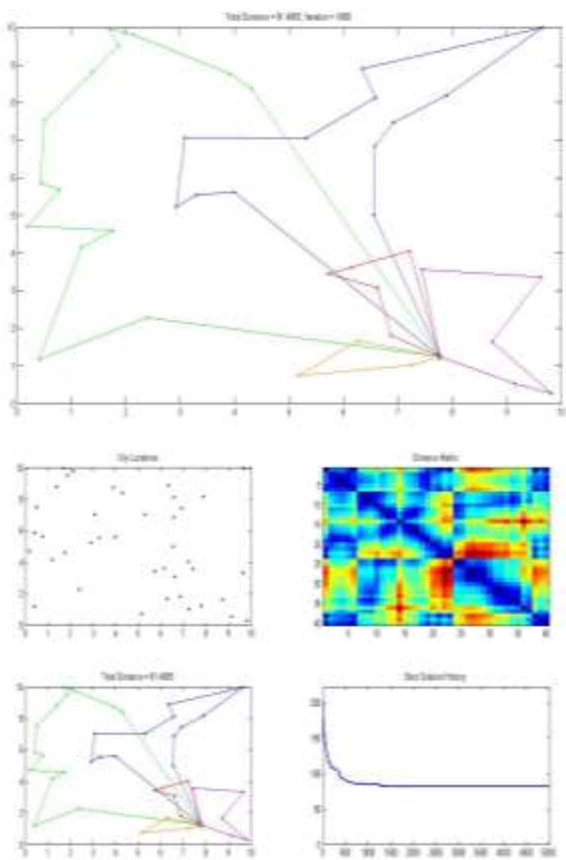


Figure 1.6: Fixed Multiple Travelling Salesmen Problem for Iteration=1680

3.1.6. Travelling Salesman Problem for Nearest Neighbour (NN) Algorithm: A particular salesman travels to each cities and completes the route by back to the city he started. Each city is visited by the salesman accurately one time.

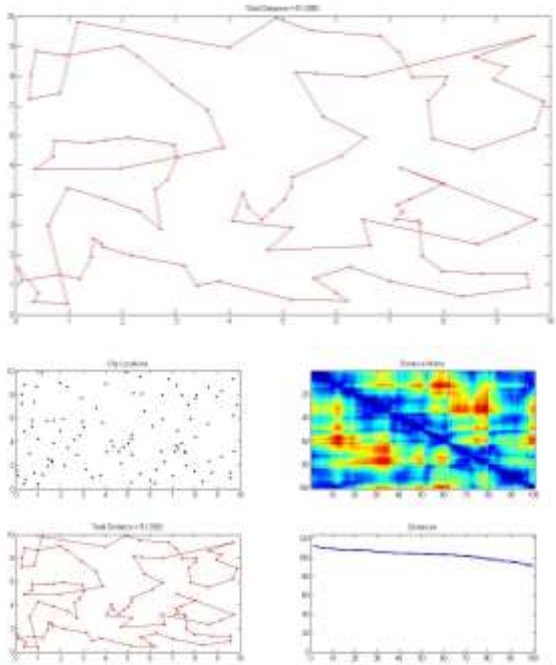


Figure 1.7: Travelling Salesman Problem for Nearest Neighbour

3.2. Result of Genetic Algorithm for Travelling Salesman Problem:

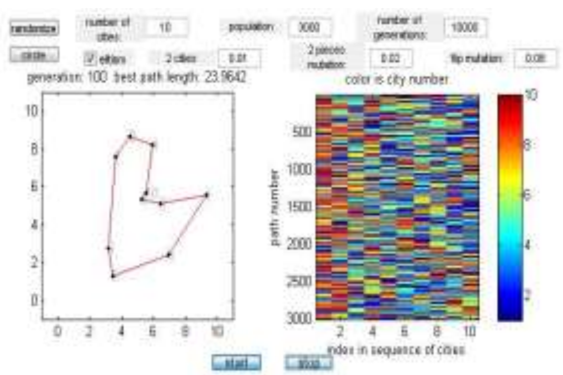


Figure 1.8: The Best Path Length for 10 cities and generation 100 is 23.9642.

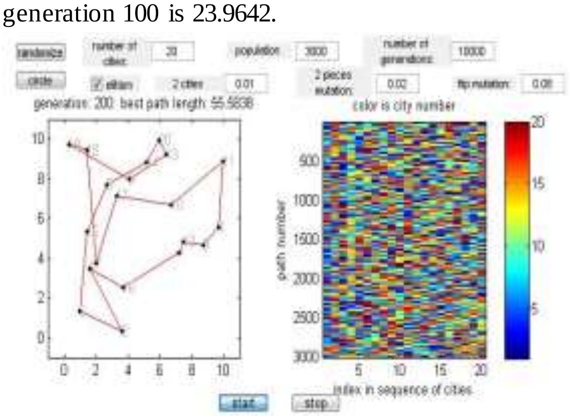


Figure 1.9: The Best Path Length for 20 cities and generation 200 is 55.58.

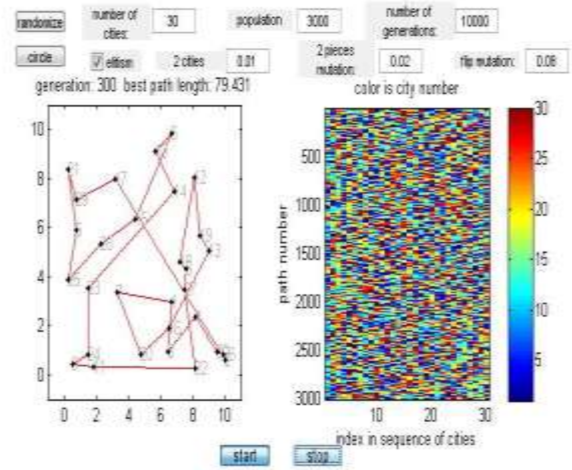


Figure 1.10: The Best Path Length for 30 cities and generation 300 is 79.43.

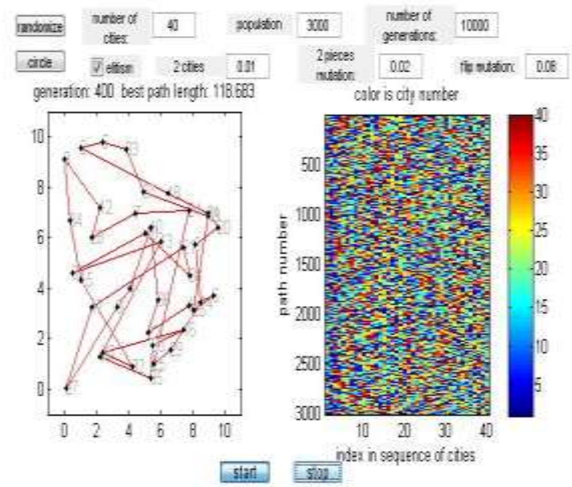


Figure 1.11: The Best Path Length for 40 cities and generation 400 is 118.68.

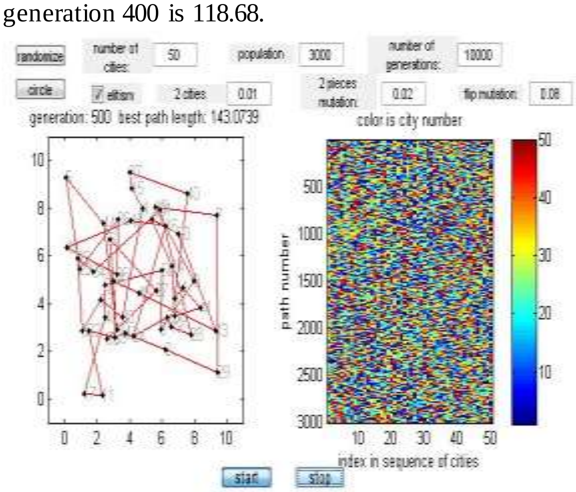


Figure 1.12: The Best Path Length for 50 cities and generation 500 is 143.07.

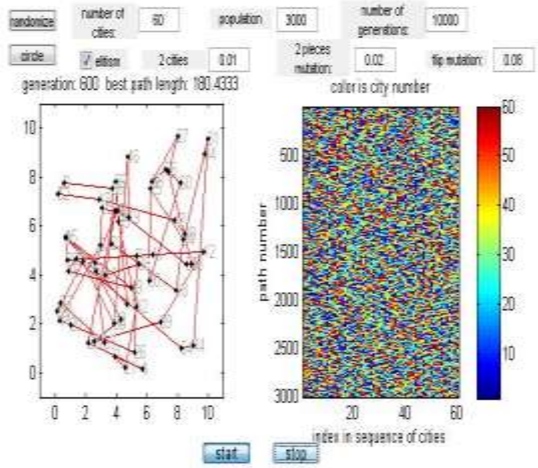


Figure 1.13: The Best Path Length for 60 cities and generation 600 is 180.43.

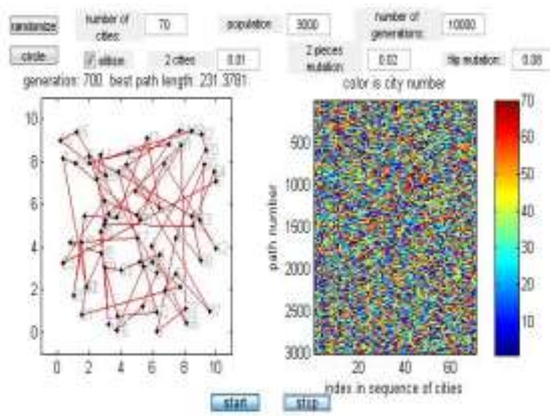


Figure 1.14: The Best Path Length for 70 cities and generation 700 is 231.37.

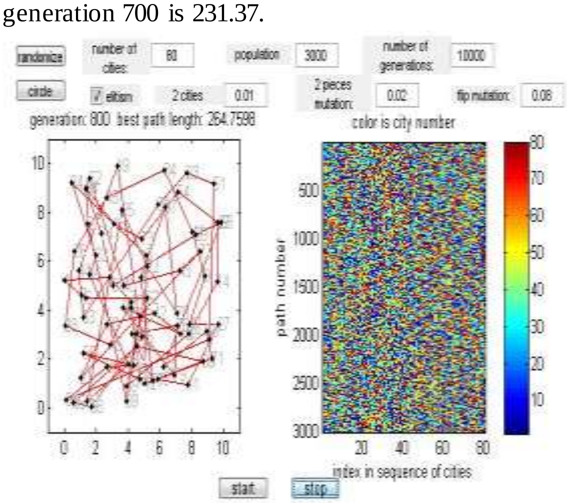


Figure 1.15: The Best Path Length for 80 cities and generation 800 is 264.75.

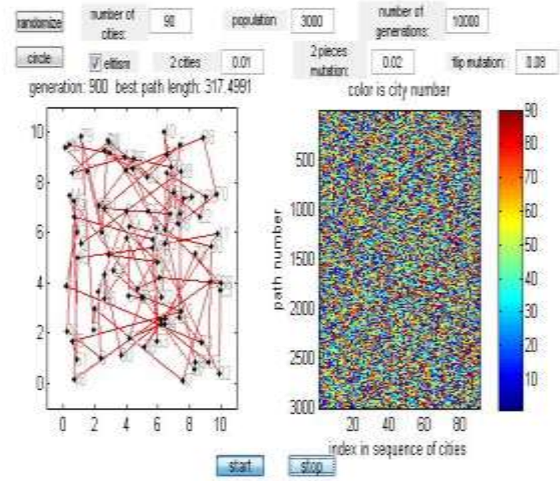


Figure 1.16: The Best Path Length for 90 cities and generation 900 is 317.49.

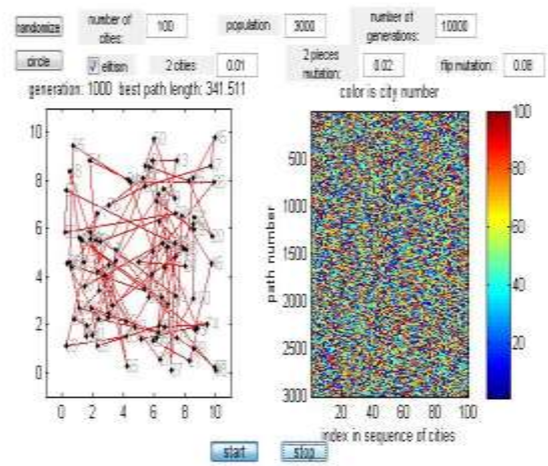


Figure 1.17: The Best Path Length for 100 cities and generation 1000 is 341.51

Table: 1.1Comparison of Best Path Length

No. of Cities	Population Size	No. of Generations	Best Path Length
10	3000	100	23.96
20	3000	200	55.58
30	3000	300	79.43
40	3000	400	118.68
50	3000	500	143.073
60	3000	600	180.43
70	3000	700	231.37
80	3000	800	264.75
90	3000	900	317.49
100	3000	1000	341.51

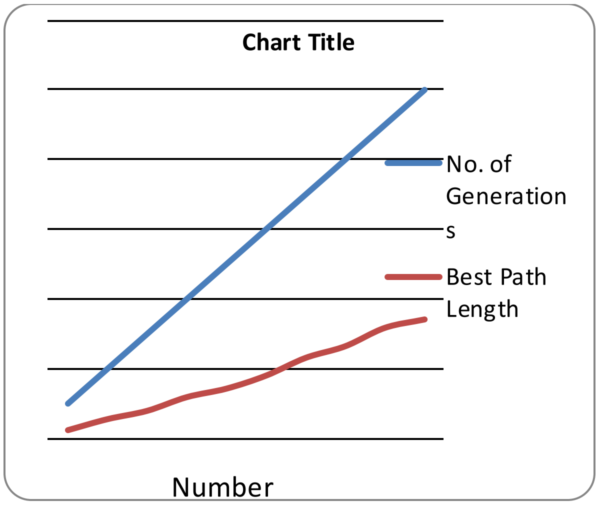


Figure 1.18: Best Path Length No. of Generation V/S No. of Cities

3.3 Result of Simulating Annealing for Travelling Salesman Problem:

A basic simulated annealing for different cities travelling salesman problem from TSPLIB2 archive.

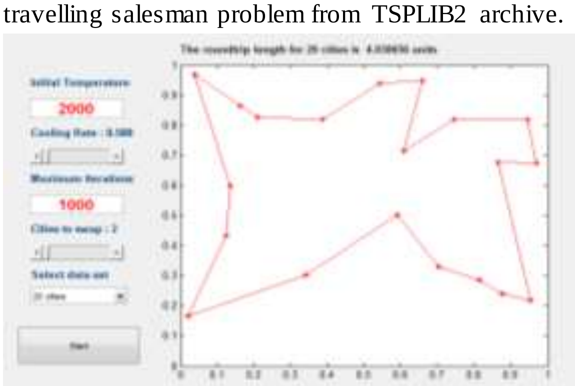


Figure 1.19: The round trip for minimal total length for 20 cities is 4.030656 units.

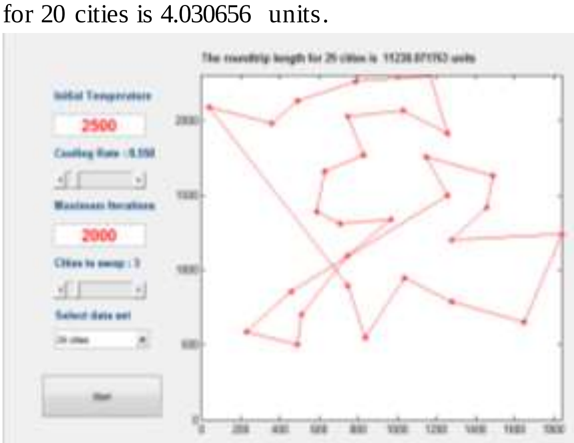


Figure 1.20: The round trip for minimal total length for 29 cities is 11238.07176 units.

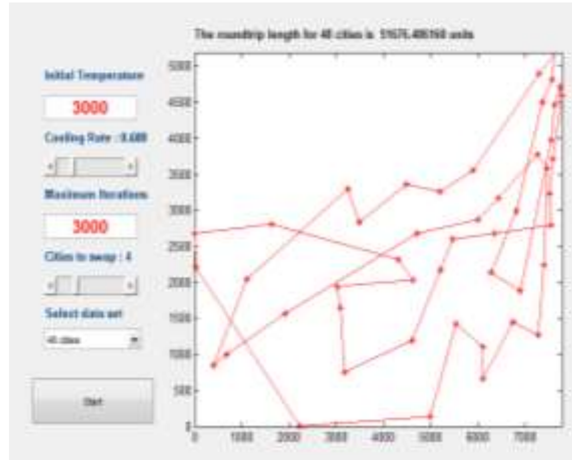


Figure 1.21: The round trip for minimal total length for 48 cities is 51676.4861 units

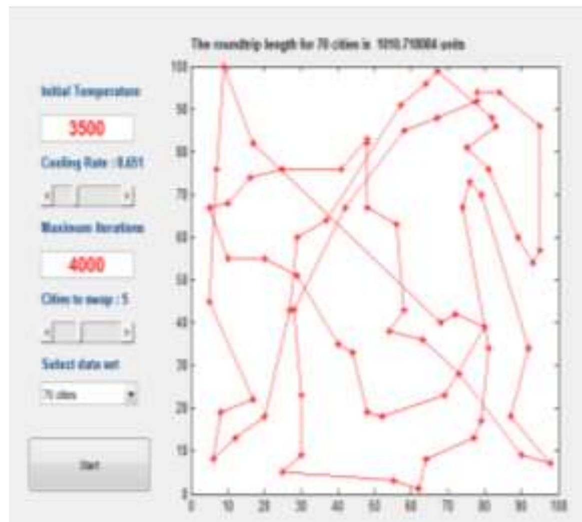


Figure 1.22: The round trip for minimal total length for 70 cities is 1010.7180 units

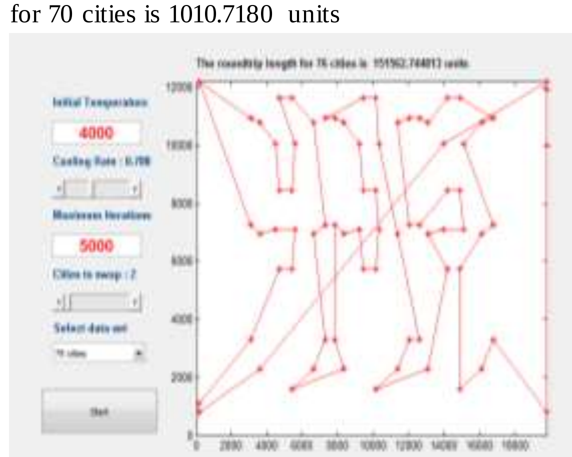


Figure 1.23: The round trip for minimal total length for 76 cities is 151562.74 units

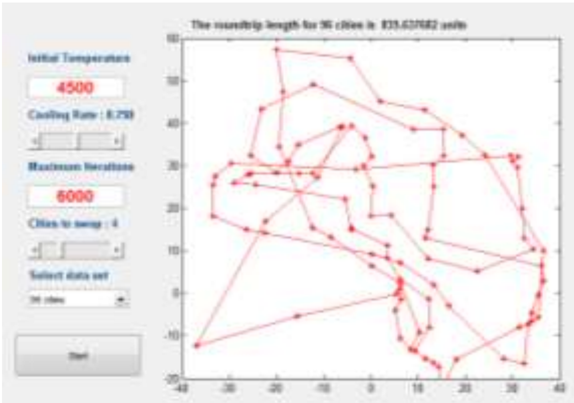


Figure 1.27: The round trip for minimal total length for 96 cities is 835.6376 units

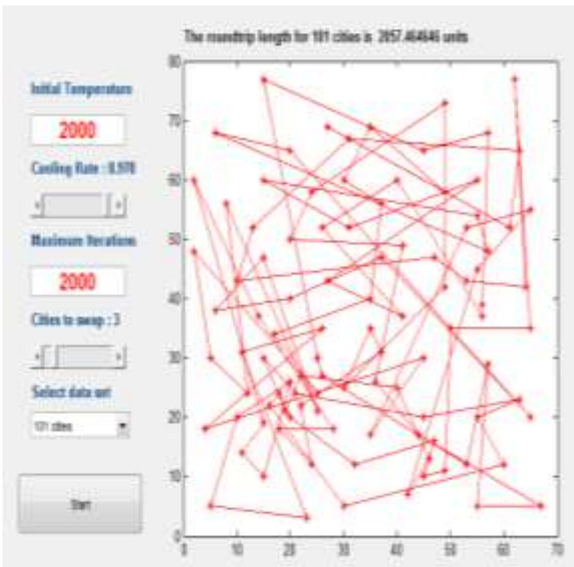


Figure 1.24: The round trip for minimal total length for 101 cities is 2057.46 units

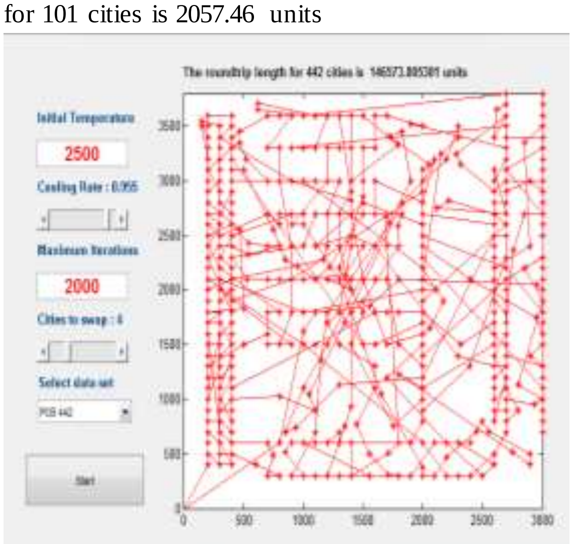


Figure 1.25: The round trip for minimal total length for 442 cities is 146573.805 units

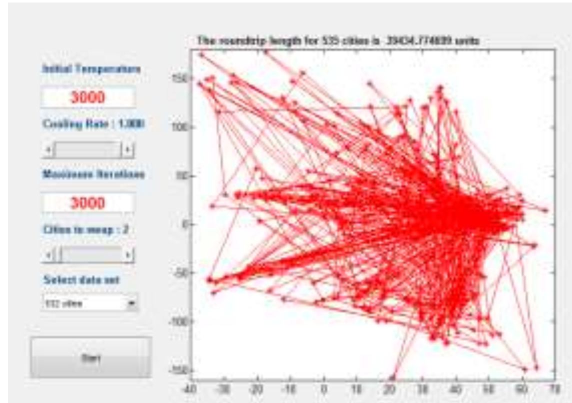


Figure 1.26: The round trip for minimal total length for 535 cities is 39434.7746 units

Table: 1.2 Comparison of Length of Round Trip (Units)

Initial Temperature	Cooling Rate	Maximum Iteration	Swap	No. of Cities	Length
2000	0.5	1000	2	20	4.030
2500	0.55	2000	3	29	11238.071
3000	0.6	3000	4	48	51678.486
3500	0.65	4000	5	70	1010.718
4000	0.7	5000	2	76	151562.74
4500	0.75	6000	4	96	835.6376
2000	0.97	2000	3	101	2057.4646
2500	0.955	2000	4	PCB 442	146573.8
3000	1	3000	2	532	39434.774

CONCLUSION

GA appears to evaluate best solutions for TSP (Travelling Salesman Problem). However it depends very much on the way how problem is encoded and crossover and mutation techniques are used. It seems that techniques uses heuristic information or encode the boundaries of the tour perform the best and give good quality indications for upcoming work in this area. Genetic algorithms have not found a better solution to the travelling salesman problem that is already known. However many of the already known best solutions have been found by some genetic algorithm method.

Simulated annealing has a potential of finding an optimal solution provided enough time is given for the annealing process. Definitely a simple concept of annealing has significant effect on finding a best solution, thus simulated annealing is indeed a good motivation for extra heuristic algorithms.

FUTURE SCOPE

In the present research work, we have implemented the combination of Genetic Algorithm & Simulated annealing for Travelling Salesman Problem to discover the shortest path and execution time between different Cities.

Combination of two between compared approaches (SA & ACO) is suggested. Whereas ACO give the best shortest distance and SA saves time in execution. So they complement each other and cancel out their own limitations

REFERENCES

[1] Goldberg, D. E. (1989). “Genetic algorithms in search optimization and machine learning”.

[2] Addison Wesley Davis, L. (ed.) (1991). “Handbook of Genetic Algorithms: Van Nostrand Reinhold”

[3] Gunnels, J., Cull, P. & Holloway, J. L. (1994). “Genetic Algorithms and Simulated Annealing for Gene Mapping”. Florida: IEEE.

[4] P. Larran˜ Aga, C.M.H. Kuijpers, R.H. Murga, I. Inza and S. Dizdarevic “Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators” 1999

[5] Kylie Brant, Aurther Benjamin. “Genetic Algorithm and Travelling Salesman Problem” December 2000.

[6] Prof. Daniel Graupe & Ritesh Gandhi “Implementation of Travelling Salesman’s Problem Using Neural Network Final Project Report” (2001)

[7] Sachin Jayaswal 2004.” A Comparative study of Tabu Search and Simulated Annealing for Travelling Salesman problem”.

[8] Karova, Milena N., Smarkov, Vassil J., and Penev, Stoyan. “Genetic operator crossover and mutation in solving the TSP problem”(2005).

[9] T.D.G wiazda, “Genetic algorithms reference: Crossover for single-objective numerical optimization problems”, Berlin: Springer Press, 2006.

[10] Laporte, Gilbert, 2006, “a short history of traveling salesman problem”.

[11] S. Sivanandam and S. Deepa, “Introduction to genetic algorithms”, Springer, 2007.

[12] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi. “Optimization by Simulated Annealing”. Computer Engineering Research Group, 2007

[13] Sze, San Nah and Tiong, Wei King. “A Comparison between Heuristic and Meta-Heuristic Methods for Solving the Multiple Travelling Salesman Problems” (2007).

[14] Ali Hamdar 2008. “Simulated Annealing Solving the Travelling Salesman Problem”.

[15] Y.H. Liu, "Diversified local search strategy under scatter search framework for the probabilistic travelling salesman problem", 2008, pp. 332-346.

[16] Paulo Henrique Siqueira, SérgioScheer& Maria TeresinhaArns Steiner “A Recurrent Neural Network to Traveling Salesman Problem” (2008)

[17] S.N.Sivanandam, S.N.Deepa “Introduction to Genetic Algorithms” Springer-Verlag Berlin Heidelberg 2008

[18] Aybars U., Serdar K., Ali C., Muhammed C. and Ali A 2009. “Genetic Algorithm Based Solution of TSP on a Sphere”

[19] Omar M. Sallabi, Younis El Haddad “An improved Genetic Algorithm to Solve the Travelling Salesman Problem [2009]”

[20] Naef Taher Al Rahedi and Jalal Atoum, “Solving TSP problem using New Operator in Genetic Algorithms” [2009].

[21] Fan Yang 2010. “Solving Traveling Salesman Problem Using Parallel Genetic Algorithm and Simulated Annealing”.

[22] Oliviu Matei et al,” An Efficient Genetic Algorithm for Solving the Generalized Traveling Salesman Problem”2010 IEEE

[23] Yang Yi et.al ,” The Improved Hybrid Genetic Algorithm for Solving TSP Based on Handel-C”, 2010 IEEE

[24] Zar Chi Su SuHlaing& May Aye Khine“Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm” (2011)

[25] Zakir H. Ahmed “Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator” (2011)