SIMULATED ELECTRICAL NETWORK APPROACH (SENA) TO HARD OPTIMIZATION PROBLEMS

Himanshu Dutt Sharma
●
Department of Electrical & Electronics Engineering,
Birla Institute of Technology & science, Pilani, (Raj.) INDIA.333031
E-mail:hdsharma@bits-pilani.ac.in

Bangale Shreyas Madhukarao
●
Department of Mechanical Engineering,
Birla Institute of Technology & Science, Pilani, (Raj.) INDIA.333031
E-mail:f2001413@bits-pilani.ac.in

ABSTRACT

A novel method is proposed for hard optimization type of problem wherein an exact optimal solution is increasingly difficult in terms of run time and memory requirements. Especially for the cases when search graph has higher number of nodes and more number of paths, which increase as factorial of node number. This is based on Simulated Electrical Network Approach (SENA) proposed here, in which the graph is modeled as an electrical network and current distribution is found which is used as a directive for search decisions. The proposed algorithm results in an approximate method that achieves average accuracy of 99.89% to reach close to the most optimal path that is found by ranking all possible paths. Conversely, it can eliminate on average 99.89% paths in polynomial time from consideration if one requires finding the most optimal one.

Key Words: Optimization, NP-Hard, TSP, Shortest path.

1 INTRODUCTION

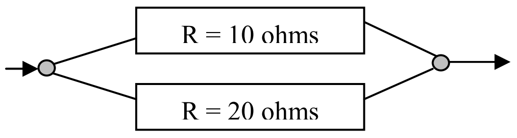
Because of the intrinsic difficulty in finding polynomial time exact solution algorithms for NP-hard optimization problems, the research has moved to the approximate solutions to these problems and development of approximate algorithms is direction of research. Some of these aim at fast approximate optimization. Traveling salesman's problem is a classical problem of this class.

There is a lot of work going in the area of optimization and planning of shortest path. The shortest-paths problem involves a weighted, possibly directed graph described by the set of edges and vertices $\{E, V\}$. Given a source vertex, s, the goal is to find the shortest existing path between s and any of the other vertices in the graph. There are two types algorithms proposed depending upon the programming involved i.e. sequential and parallel. These types of problems involve weighted graphs and can be applied to Euclidean or non-Euclidean cases. Among these TSP (Traveling Salesman's Problem) stands as one of the most difficult and sought after problem and has remained a challenge for many algorithm planners and also serves as a problem for testing optimization algorithms efficiency. There are many approximations to solve this problem, as any polynomial time algorithm does not find exact solution yet [1]. The approximation algorithm using the triangle inequality is well known developed by Christofides [2]. Artificial intelligence based techniques are also developed to search for optimal paths e.g. genetic algorithms, simulated annealing, and neural nets are examples of these [3]. Some Other attempts to solve TSP include generalization in which, for each city, a neighborhood is specified in which the salesperson can meet the client is also approximable for a variety of neighborhood types such as unit segments, unit circles, and unit rectangles [4]. Another generalization in which the salesperson has to rearrange some objects while following the route is approximable within 2.5 [5]. A prize-collecting variation in which a penalty is associated with each vertex and the goal is to minimize the cost of the tour and the vertices not in the tour [6]. A variation in which vertices can be revisited and the goal is to minimize the sum of the latencies of all vertices, where the latency of a vertex c is the length of the tour from the starting point to c, is approximable within 29 and is APX-complete [7]. A combination of this problem and the matching problem, also called Printed Circuit Board Assembly, is approximable within 2.5 [8]. Finally, the variation in which a Hamiltonian path is looked for rather than a tour is also approximable within 1.5 in the general case while if both end vertices are specified it is approximable within 5/3 [9].

In the proposed Simulated Electrical Network Approach (SENA), to prove that it indeed gives paths in optimal range, some of the most complex sets of input graph are taken i.e. fully connected, non-Euclidean; require traversing all nodes once and only once. The test instances were randomly generated. This is similar to solving a TSP instance in terms of complexity. Cases are tested for varying number of nodes. In these types of graphs, the exact search techniques show exponential complexity for the algorithms, and therefore cannot be applied for the cases with large node numbers. Whereas this technique reduces time complexity to polynomial time and can be used as a good approximate technique for optimization. For large number of nodes its average performance returns a path, which is within top 1.1% of optimal paths. Hence the method introduced in the next section proves to be very effective in producing an approximate solution to the shortest path type problem in a polynomial time.

2 SIMULATED ELECTRICAL NETWORK APPROACH: ANALOGY WITH CURRENT FLOW

In this approach, at first the cost of each link of a given search graph is modeled as a branch resistance of an electrical network. A voltage source is added between pre-specified START and DESTINATION nodes and then current distribution is found in the transformed electrical network. In this electrical network model, the new approach relies on the observation that current flow in an electrical network follows a fundamental rule: maximum amount of current tries to take minimum resistance path, which is the key to eliminate most of the paths which do not come anywhere close to the least resistive (cost) ones. Following this common observation one can bring a sort of foresight in the network for path search type problems. To get the substance of the approach in its simplest form, consider the following simple network's example:



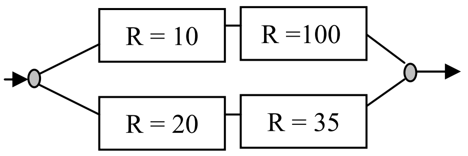


Figure_1 & 2

a) The current division in figure_1 is such that the larger current passes through the lesser resistance path. Therefore a decision can be made at node-1 of choosing a link, which has highest current flowing among all possibilities from that node to other nodes. This in effect equips a path planner with a foresight.

b) The Figure_2 is a modified case of (a). Although the resistances connected to node number '1' remains the same but at a later node, the introduced resistances change the situation in a way that the overall cost of the earlier lesser-cost path is now high but the connections as seen from the first node remain the same. To a simple path planner it becomes necessary to scan the whole set of links coming in path, otherwise it will misguide and a local minimum situation is most likely to be achieved. Whereas a current flow based approach would be a better decision maker in such cases. It will still give the correct path through R=20,35 part of the circuit. A full appraisal of proposed method in large fully connected networks is done in following sections.

3 ALGORITHM FOR SENA

This approach is applicable to both Euclidean and Non-Euclidean type of graphs, as the costs are not chosen on Euclidean basis. The algorithm is tested for fully connected, non-Euclidean; require traversing all nodes once and only once before reaching to a pre-specified destination. Here, It is to be noticed that these type of problems are as hard as TSP. The approach involves three major steps:
- Modeling the given graph in Electrical circuit;

- Solving the modeled electrical network for currents in each branch;
- Simple decision making by looking at current-magnitudes from each node;

And a straightforward very close solution to the exact shortest path is achieved.
The detail procedure adopted is as follows:

- Generate random instances of graphs;
- Convert cost to resistance i.e. $R_{ij}= C_{ij}$; where i & j are nodes
- Identify the source node as start point, make it positive terminal of battery;
- Identify the destination node make it a negative terminal of battery;
- Specify a voltage and solve it for current in each branch;
- From each node go to the next node to which maximum current flows;
- Destination node is traversed only in the last step.

A path thus found is indeed very close to exact optimal. See table figure_3.

4 RESULTS

In the graph shown in figure_3, we generated as many as 25 different fully connected graph and exact solutions for each node number, varying the number of nodes as N = 6,7,8,9,10,11,12,13. The search performed for shortest path had to get a path from possible number of paths ranging as few as 24 for N=6 to as many as 39,916,800 for N=13.

Table1: % Optimization Achieved for Number of Nodes in Graphs

Trials	Node 6	Node 7	Node 8	Node 9	Node 10	Node 11	Node 12	Node 13
1	73.91	94.96	60.64	99.27	84.37	98.4	99.96	99.27
2	73.91	96.64	91.38	98.47	95.19	99.91	99.86	99.99
3	69.56	65.55	90.33	94.30	99.91	99.99	99.99	99.96
4	78.26	88.24	94.30	96.49	83.80	99.39	99.98	99.99
5	91.30	96.64	96.11	99.76	83.89	97.77	99.99	99.99
6	82.61	95.80	97.36	90.67	96.28	96.48	99.83	99.76
7	**100**	44.54	97.08	98.33	98.83	97.97	99.41	99.99
8	82.61	**100**	98.89	99.80	99.68	99.29	99.93	99.98
9	60.87	86.55	98.47	77.40	96.38	99.99	99.98	99.97
10	95.65	94.96	74.55	92.94	99.78	99.99	96.89	99.99
11	**100**	95.80	98.05	**100**	99.72	99.97	99.83	99.99
12	86.96	**100**	94.02	96.47	93.33	99.99	99.81	99.66
13	82.61	**100**	79.00	96.57	99.91	99.99	99.99	99.97
14	30.43	97.48	93.60	75.00	97.80	99.82	99.94	99.51
15	**100**	**100**	80.67	99.44	93.34	99.66	99.48	99.92
16	**100**	93.28	90.33	99.98	99.73	99.85	98.52	99.8
17	95.65	83.19	95.83	98.83	99.15	98.84	99.41	99.99
18	78.26	**100**	90.40	83.51	99.99	98.7	99.89	99.67
19	69.56	**100**	98.89	66.24	99.74	99.99	90.45	99.99
20	**100**	98.32	95.27	93.65	99.84	99.68	99.86	**100**
21	82.61	99.16	99.30	94.90	99.64	97.22	99.84	**100**
22	69.57	77.31	95.41	95.61	98.95	99.37	99.83	99.99
23	95.65	83.19	97.50	**100**	99.26	99.29	99.39	99.97
24	30.43	90.76	61.75	91.70	91.97	99.98	99.89	99.99
25	91.30	66.39	91.80	99.94	99.85	99.81	99.95	99.99

Figure_3

Table2: Overall Analysis of Results Figure_4

SR. NO.	No. of Nodes	No. of Samples	Total No. Of paths	Best result ranking	Average Result (Within x% of Optimal)	Variance	Average Eliminated Paths (%) $\overline{X}$
1	6	25	24	1st	19.19	18.67	80.81
2	7	25	120	1st	10.05	13.44	89.95
3	8	25	720	1st	9.33	10.62	90.67
4	9	25	5040	1st	6.46	8.68	93.54
5	10	25	40,320	14th	3.59	5.12	96.41
6	11	25	362,880	26th	0.76	0.956	99.24
7	12	25	3,628,800	176th	0.73	0.91	99.27
8	13	25	39,916,800	1st	0.11	0.181	99.89

Figure_4

The table in figure_4 and Plot of average of results converging towards the exact optimal solution in figure_5 along with standard deviation show the effectiveness of the algorithm. Following points are inferred from these,

1 As number of paths increase, the results are the bold line in graph, which refers to the steep *Average convergence* of results towards optimal solution. As seen, it is coming in close range of within 0.11 % of the most optimal path. *The range x% refers:* "the percentage of number of paths lying in between the achieved solution and the exact optimal solution". This comparison range is achieved, by finding all possibilities of paths and costs, then sorting these in increasing order for all instances.

2 The fourth column in table *best result ranking* shows that for higher number of nodes in our randomly generated instances, the best result was as close as 99.9928% to the most optimal path i.e. 26th out of 362,880 paths. 99.9979% to the most optimal path i.e. 176th out of 3,628,800 paths. For lesser number of nodes many times it found the best path i.e. 100% optimization e.g. 1st out of 720 paths. Also at random sometimes it hits best path for large number of nodes i.e. twice in case of 13 nodes.

3 The monotonous nature of the Average optimization curve and Standard deviation curve in figure_5 shows that there is natural tendency of this algorithm to converge towards optimal result as the number of nodes increase.

4 The nature of small and reducing standard deviation reveals that the samples had smaller and smaller deviation in their average performance upon increasing the node numbers.

5 The last column of table_2 shows elimination capacity of algorithm. It is also getting better for higher node numbers, e.g. for N=13 on an average it can eliminate ~99.89% paths which are away from optimal.

6 Following graph in fig_5 clearly depicts that optimization as well as deviation nearly reaches to the best as nodes are increased. Thus hypothesis proves to be an excellent approximate algorithm for optimization in highly complex, large number of nodes, fully connected graphs of Euclidean or Non- Euclidean type.

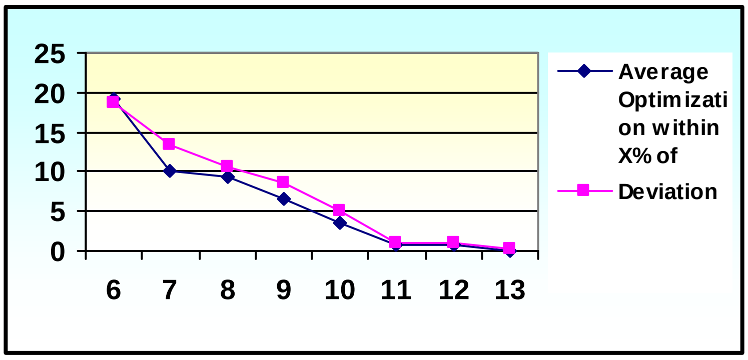


Figure_5

5 ERROR CALCULATION (ACCEPTANCE REGION OF μ)

To validate results further, the maximum error in $\overline{X}$ is calculated, given the probability as 99%. This will also give the estimate of actual population mean μ. Twenty-five samples are taken for each node number for optimization. For these 25 samples, the mean estimates of $\overline{X}$ are $\overline{X}$ = 80.81 for (nodes 6), $\overline{X}$ = 89.95 for (nodes 7), $\overline{X}$ = 90.33 for (nodes 8), $\overline{X}$ = 93.54 for (nodes 9), $\overline{X}$ = 96.41 for (nodes 10), $\overline{X}$ = 99.24 for (nodes 11), $\overline{X}$ = 99.27 for (nodes 12), $\overline{X}$ = 99.89 for (nodes 13). Therefore In our case: n = 25,P = 0.99,1-$\propto$ = 0.99,$\propto$ = 0.01 and $Z_{\propto/2}$ = $Z_{0.005}$ = 2 .575. Then Maximum error of estimate is given by the following formula. $E = Z_{\alpha/2}(\sigma/\sqrt{n})$

Figure_6

Nodes	$Z_{\propto/2}$	σ	Mean based on trials $\overline{X}$ %	E=$\lvert\overline{X}-\mu\rvert$ (probabilistic estimation in error with 99% confidence)	Lowest possible value of μ %	Highest possible value of μ %
6	2.575	18.67	80.81	9.61	71.2	90.42
7	2.575	13.44	89.95	6.92	83.03	96.87
8	2.575	10.62	90.33	5.47	84.86	95.8
9	2.575	8.68	93.54	4.47	89.07	98.01
10	2.575	5.127	96.41	2.64	93.77	99.05
11	2.575	0.956	99.24	0.49	98.75	99.73
12	2.575	0.91	99.27	0.47	98.80	99.74
13	2.575	0.181	99.89	0.09	99.80	99.98

From above graph and table figure_6&7, it is concluded that actual mean i.e. average optimization achieved in the proposed algorithm will be greater than 99% for nodes above N=10, with 99% confidence. It strongly supports the SENA's validity.

Figure_7

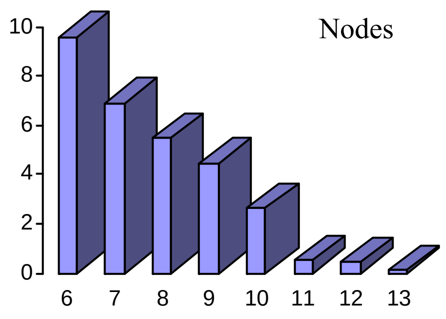
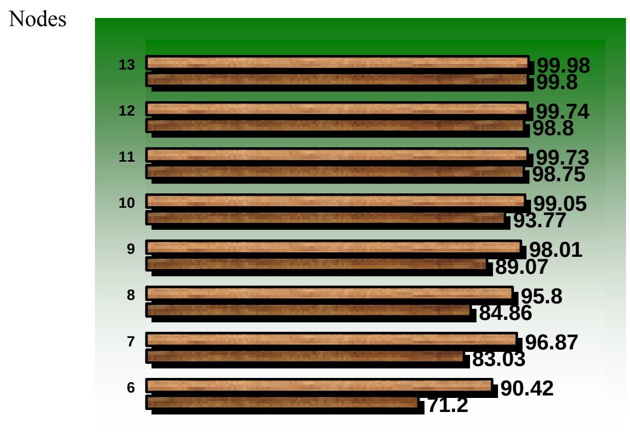


Figure_8



6 PROBABILITY OF ACHIEVING OPTIMIZATION

Given are the values of $\overline{X}$ for different nodes for a sample of 25 with variance (σ) values, following table finds probability of getting optimization between 90% to 100 % for different number of nodes using the following probability function: $F(Z) = F((\overline{X} - \mu)/\sigma * n^{1/2})$

Nodes	$\overline{X}$	μ Interval	Variance σ	Trials n	F (Z) for μ = 90	F (Z) for μ = 100	P= F (Z) $_{90}$ – F (Z) $_{100}$
6	80.81	90 - 100	18.67	25	0.0069	0.001	0.0059
7	89.95	90 –100	13.44	25	0.496	0.0001	0.496
8	90.33	90 –100	10.62	25	0.5596	0.0002	0.5596
9	94.56	90 –100	8.68	25	0.9788	0	0.9788
10	96.41	90 –100	5.127	25	0.99999	0	0.9999 ≈ 1
11	99.24	90 - 100	0.956	25	1	0	≈ 1
12	99.27	90 - 100	0.91	25	1	0	≈ 1
13	99.89	90 - 100	0.18	25	1	0	≈ 1

Figure_9

The above-tabulated results are plotted and it shows that as number of nodes increases the probability of getting optimization between the said regions is approaching 1.

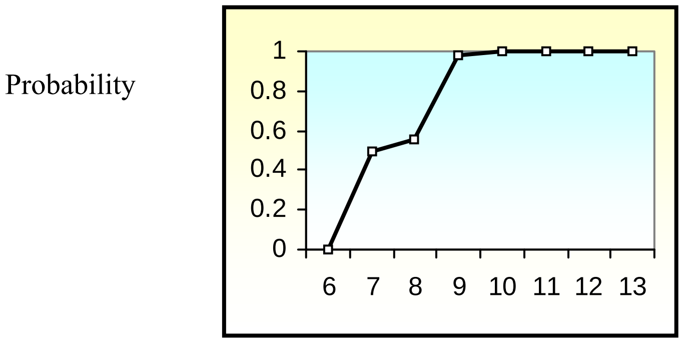


Figure_10

7 INVERSE LOGIC FOR SENA (AN ALTERNATIVE APPROACH)

An alternative to the one stated above is proposed here. **Path is searched on the basis of inverse technique where inverse values of costs are taken and path is searched for minimum current. Costs are then added normally for checking optimization.** After applying the normal probabilistic hypothesis, it came to know that inverse technique optimization increases significantly for the cases where optimization is less in normal SENA technique. Comparison is sufficient for 10 numbers of nodes and prediction can be done for further nodes on the basis of probability and statistics.

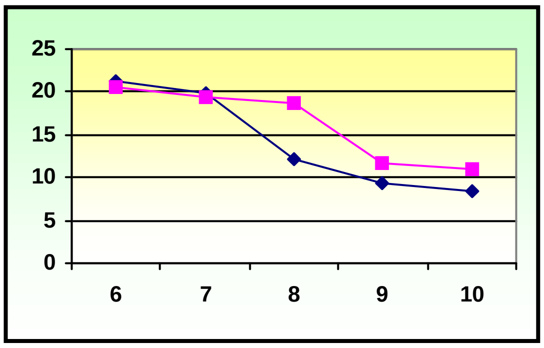
TRIALS	NODES 6	NODES 7	NODES 8	NODES 9	NODES 10
1	95.65	79.83	95.41	78.29	99.10
2	95.65	57.98	99.17	99.05	99.22
3	91.30	89.08	90.68	99.35	99.70
4	69.57	97.48	98.19	95.42	88.03
5	91.30	**100**	99.43	97.20	99.69
6	86.96	51.26	92.63	95.36	99.94
7	95.65	98.32	64.39	95.48	95.92
8	82.61	95.80	88.48	71.18	99.94
9	73.91	82.35	95.13	73.69	96.58
10	65.22	83.19	97.07	99.40	56.34
11	56.52	94.96	95.83	99.01	99.34
12	26.09	98.32	92.49	88.27	94.11
13	82.61	97.48	90.82	57.93	92.39
14	56.57	96.64	92.91	85.29	98.76
15	95.65	79.83	66.90	99.62	92.39
16	95.65	80.14	34.63	97.50	99.74
17	56.52	96.64	98.61	99.82	87.25
18	86.96	78.15	99.86	94.17	98.59
19	60.87	97.48	**100**	87.34	91.20
20	**100**	85.71	96.49	94.43	98.31
21	95.65	70.59	99.58	98.79	64.04
22	86.96	99.16	99.72	65.59	88.30
23	95.65	87.39	28.93	99.72	99.81
24	30.43	21.08	90.13	97.62	96.94
25	95.65	60.50	89.85	95.10	91.66

Figure_11: % Optimization Achieved for Number of Nodes in Graph

Figure_12: Overall Analysis of Results

Nodes	Optimization%	Deviation σ
6	78.77	20.48
7	80.14	19.38
8	87.89	18.77
9	90.58	11.69
10	91.66	10.89

Figure_13



Curve clearly depicts that optimization as well as deviation nearly reaches to 10% as we further increase the nodes. PINK line denotes efficiency while BLUE line indicates deviation. Thus hypothesis proves better.

8. APPLICABILITY OF INVERSE TECHNIQUE FOR OPTIMIZATION

At first, It looks as if inverse technique is not so useful because of its inconsistent results. But it is quite useful when acyclic technique is giving bad results in terms of optimization. 25 trials were taken and graphs for various nodes were plotted. It is concluded that when acyclic graphs were giving depressions for particular regions, at that time inverse was at its peak i.e. better optimizing. For example, we have given the visual reference to it in the following figure_14, 15, and 16. Following Charts show comparison of trials between normal and inverse technique (an example is taken for 9 nodes) as well as distribution about superiority of inverse and acyclic techniques over each other for similar 25 trials. Pointer shows for example; how inverse succeeds over normal SENA for a given case.

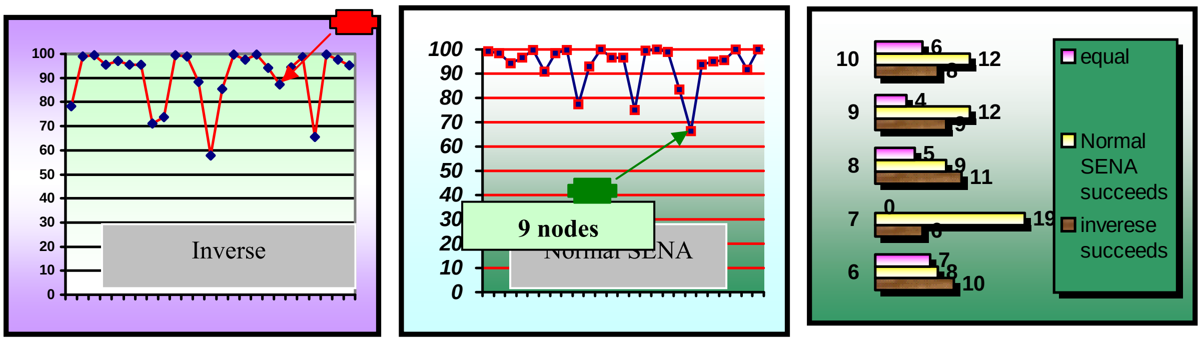


Figure_14, 15, 16

9 CYCLIC NETWORK

Algorithm for cyclic network is given below.
1. Take node input from user.
2. Start and goal destinations are the same.
3. For programming purpose, develop a pseudo goal node.
4. Resistance between all nodes and pseudo goal node will be of the order of the resistance value between start node and all other corresponding nodes.
5.Now same logic is implemented as that of acyclic network to find the path.
6.At last, pseudo goal node is removed from the obtained path and in that place; start node is kept for the final path.

Since the trend shown by the results of cyclic graphs match that of acyclic graph optimization pattern, therefore results over nine nodes are avoided here.

9.1 RESULTS

Trials	Node 6	Node 7	Node 8	Node 9
1	94.958	90.542	98.591	89.107
2	78.992	96.801	92.102	98.646
3	81.513	89.847	82.616	99.807
4	65.546	71.21	97.5	99.529
5	86.555	99.722	99.861	99.747
6	99.16	99.583	93.848	99.571
7	**100**	**100**	97.321	99.722
8	82.353	97.497	99.623	99.893
9	99.16	97.775	98.809	84.824
10	**100**	91.099	99.782	99.606
11	90.756	98.609	99.861	99.98
12	**100**	**100**	92.499	99.931
13	98.319	97.357	95.872	98.79
14	**100**	99.166	99.008	98.676
15	99.16	99.305	**100**	99.576
16	98.319	83.032	97.023	**100**

Trials	Node 6	Node 7	Node 8	Node 9
17	98.319	98.609	99.028	98.971
18	91.597	96.801	95.138	93.219
19	83.193	94.159	94.245	97.433
20	91.597	86.787	96.229	92.52
21	**100**	99.583	99.544	92.969
22	82.353	91.377	98.234	97.133
23	**100**	91.377	58.821	99.812
24	99.16	69.68	90.375	99.955
25	94.118	99.722	95.197	99.762

Figure_17 % Optimization Achieved for Number of Nodes in Graphs

Nodes	Optimization%	DEVIATION σ
6	92.605	9.1
7	93.586	8.37
8	94.845	8.48
9	97.567	3.95

Figure_18: Overall Analysis of Results

9.2 PROBABILITY OF ACHIEVING OPTIMIZATION

Nodes	$\overline{X}$	μ Interval	Variance σ	Trials n	F (Z) for μ = 90	F (Z) for μ = 100	P= F $(Z)_{90}$ – F $(Z)_{100}$
6	92.605	90 - 100	9.1	25	0.9236	0.00003	**0.9236**
7	93.586	90 –100	8.37	25	0.9838	0.00001	**0.9838**
8	94.845	90 –100	8.48	25	0.9979	0.0012	**0.9967**
9	97.567	90 –100	3.95	25	1	0.001	**≈ 1**

Figure_19 Probability Estimation

The above-tabulated results show that as number of nodes increase, the probability of getting optimization between the said regions is approaching 1. It is faster than what was observed in the acyclic graphs.

10 CONCLUSION

It has been shown in the results that the proposed SENA-method is capable of returning a near optimal solution for shortest path finding type of problems. Thus it can handle hard optimization for the problems that have complexity of fully connected graphs where the number of possible paths increase in proportion to factorial of number of nodes in the graph. Also this technique is applicable to Euclidean or non-Euclidean cases equally well as there are no constraints of Euclidean geometry assumed in the formation of graph instances. Also it is established that its elimination capacity for paths which will be close to optimal, from all possibilities is reaching to ~99% on average basis for higher number of nodes where other techniques starts reducing their efficiency, in contrast, its in fact monotonously showing better results. The statistical analysis shows using probabilistic estimate that as number of nodes increases; the near complete optimization can be achieved. The cases where SENA fails to achieve required optimization; inverse technique can be used as an efficient tool. So conclusively it's quite effective for determining a close to optimal heuristic. Further, it is shown that the technique is equally applicable to cyclic graphs also. *Therefore the SENA algorithm is capable of returning result which is nearly the best optimization in graphs of varying nature: cyclic, acyclic, Euclidean, non-Euclidean and of the highest order of complexity i.e. fully connected wherein the total paths are increasing in proportion to factorial of node number.*

11 REFERENCES

1. Cormen, Leiserson and Rivest, Introduction to Algorithms, McGraw-Hill, 1994.
2. N.Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Symposium on New Directions and Recent Results in Algorithms and Complexity, page 441, New York, NY, 1976. Academic Press.
3. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, New York, 1992.
4. Arkin, E. M., and Hassin, R. ``Approximation algorithms for the geometric covering salesman problem'', *Disc. Appl. Math.* **55**, 197-218. (1994),
5. Anily, S., and Hassin, R. ``The swapping problem'', *Networks* **22**, 419-433. (1992),
6. Goemans, M. X., and Williamson, D. P. ``A general approximation technique for constrained forest problems'', *SIAM J. Comp.* **24**, 296-317. (1995a),
7. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., and Sudan, M, ``The minimum latency problem'', *Proc. 26th Ann. ACM Symp. on Theory of Comp.*, ACM, 163-171 (1994).
8. Michel, C., Schroeter, H., and Srivastav, A., ``TSP and matching in printed circuit board assembly'', *European Symposium on Operations Research*. (1995).
9. Hoogeveen, J. A. ``Analysis of Christofides heuristic: Some paths are more difficult than cycles'', *Oper. Res. Lett.* **10**, 178-193 (1978).