International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Exploring a Minimum Cost Solution for Traveling Salesman Problem using Parallel Simulated Annealing

Geerisha Jain[1], Dr. Anto S[2], Dewang Mehta[3]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, India

[2]Professor, Department of Computational Intelligence, Vellore Institute of Technology, India

[3]School of Computer Science and Engineering, Vellore Institute of Technology, India

Corresponding author (s) E-mail(s): [1]geerishajain1[at]gmail.com; [2]Contributing Authors: anto.s[at]vit.ac.in
[3]dewang_mehta[at]hotmail.com

Abstract: ***In this paper, we extend the traditional Simulated Annealing (SA) algorithm to provide a modified version with incorporated parallel processing. The algorithm is used to solve the NP-hard Traveling Salesman problem in which there is a specified map of cities, and the algorithm aims to discover the shortest and the most optimized route that begins at one point, travels once through all the cities, and then returns to the initial point. The main objective is to discover the likelihood of having a zero-cost path with n cities and p processors specifically where they run in parallel. To achieve this analysis, simulation was performed, and the results obtained proved that the proposed method showed promising results in terms of finding a zero-cost path in lesser execution time when running for a large number of processes on parallel processors.***

Keywords: Travelling Salesman Problem, TSP, Simulated Annealing, minimum cost solution, optimization, Parallel Computing

1. Introduction

The traveling salesman problem (TSP) is known to be a typical NP-Hard problem in combinatorial optimization and a significant situation when it comes on to the graph theory and computational perspective. In this problem, a distinct number of cities and the distances every city has from each other is given. The objective is to discover the shortest optimized possible path that makes sure that every city has been visited once and in the end, it comes back to the initial one. Apart from this, it is also important that the distances between the cities are known. In the software world, this problem is frequently seen as a graph problem that makes use of a symmetric adjacency matrix in which every node represents the city and edges represent the cost or distance (Andreas Björklund, 2012)

Countless attempts at finding a solution to TSP have been made using various algorithms, mathematical models, and optimization searches. Most existing literature, dealing with TSP is based on Tabu Search Implementation (Sumanta Basu 2008, Basu S 2012) owing to the norm that Tabu search is one of the most widely applied metaheuristics for solving TSP, but Tabu search has its own research gaps that are yet to be explored. However, its variations and separate versions provide solutions to a vast number of computing problems.Ant Colony Optimization (Hingrajiya KH 2012) and Bee colony algorithm (Anshul Singh 2012) with its basic mechanism of bees foraging behavior and its efficiency in solving the shortest path among various routes have been implemented. Neighborhood search is useful when exploitation is desired. It can be applied after every bee cycle to enhance the quality of solutions. Evolutionary algorithm (Huai-Kuang Tsai 2004) described new crossover operators and mutation operators suitable for solving TSP by Genetic Augmentation with the availability of fast computing facilities, this search technique can be utilized to solve TSPs with large dimensions. Studies have also been carried out extensively on the genetic algorithm (Naveen Kumar 2012, Gupta S 2013, Khattar S 2014) due to its success rate in NP-hard problems, however, it depends highly on the system in which the problem is encoded, and which crossover and mutation techniques are used.

With regards to the simulated annealing (SA) algorithm, comparatively lesser research and variations have been explored to date. A new simulated annealing algorithm was proposed, called a list-based simulated annealing algorithm (Zhan et al.2016), in order to solve the traveling-salesman problem. Experimental results indicated that the proposed algorithm had competitive performance compared to the other algorithms. Sometime later, a hybrid simulated annealing algorithm based on tabu search (Lin, Y 2016) was proposed to solve TSP. Experimental results demonstrated that the proposed algorithm improved the accuracy and efficiency of the traditional SA. In the line with that, a simulated annealing algorithm based on symbiotic-organism search (Ezugwu2017) was presented to better solve the TSP. Comparative analysis revealed that the proposed algorithm had advantages with regard to factors such as convergence, average execution time, and percentage deviations. In an attempt to further optimize and aim for a more efficient solution, an improved simulated annealing algorithm (Zhao, D 2017) was devised. Numerous efforts on searching for a zero-cost path have been made, with each approach having its own solution sets and limitations.

This note aims to extend the traditional Simulated Annealing (SA) algorithm by incorporating parallel computing to find a

solution to the Travelling Salesman problem. The remaining part of the paper is organized as below: In Section 2, we give an overview of the problem statement and the intended solution; discuss the mathematical representation of the traveling man problem; the idea behind simulated annealing, its acceptance criterion; propose algorithms for the enhanced new version of simulated annealing with parallel processing, and give in-depth coverage of the model's architecture, operations involved, and parameters defined. Section 3 of the paper covers the results obtained from the simulation of the algorithm when run in parallel. Finally, Section 4 presents the conclusion of the devised algorithm and gains insights into the viability of the solution.

2. Methodology

We have a discrete space of cities, and the algorithm finds the shortest route that starts at one of the towns, goes once through every other point, and returns to the first one as shown in Figure 1 (Zhan et al. 2016). The main goal is to explore the possibility of having a zero-cost solution with n cities and p processors running in parallel. To perform this, we are using a TSP algorithm and making it run in parallel. The algorithm devised is for Parallel SA to explore the possibility of having a zero-cost solution with n cities and p processors.

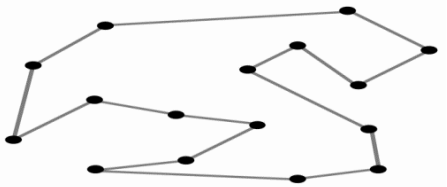

Figure 1: Travelling Salesman Problem connecting n cities asymmetrically

2.1 Travelling Salesman Problem (TSP)

TSP can be demonstrated as an undirected weighted graph, such that the graph's vertices represent the cities, the graph's edges represent the paths, and an edge's weight depicts the path's distance. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once (Mijwil2016). In the majority of the cases, the model is a complete graph (such that each pair of vertices is connected by an edge). If no route exists between two cities, adding a sufficiently long edge will result in completing the graph, not affecting the optimal tour at the same time (Andreas Björklund 2012)

We label the cities with the numbers 1, …, n and define the constraints given by (1)

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Take $c_{ij} > 0$ to be the distance from city i to city j. Then TSP can be written as the following integer linear programming problem as given by (2), (3), and (4)

$$\text{Min} \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij}\, x_{ij} \quad (2)$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j=1, \ldots, n; \quad (3)$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i=1, \ldots, n; \quad (4)$$

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} \leq |Q| - 1 \qquad \forall Q \subsetneq \{1, \ldots, n\}, |Q| \geq 2 \quad (5)$$

The last constraint of the formulation given by (5) ensures no proper subset Q can form a sub-tour, so the solution returned is a single tour and not the union of smaller tours. Since this leads to an exponential number of possible constraints, in actuality it is solved using row generation.

2.2 Simulated Annealing (SA)

SA is basically referred to as a probabilistic method to estimate the global optimum for a given processor method. This process discovers the Temperature concept and agrees on worse solutions as it explores the solution space. This examination of poor solutions is a major feature of this meta-heuristic approach (Kirkpatrick 1983). Occasionally it is made to behave as equivalence to a mountain: simulated annealing admits larger distances reaching high onto the mountain of solutions and states if we overcome the hill peak, we reach the best solution which is optimized.

2.3 Acceptance Criterion for SA

Simulated Annealing (SA) is inspired by a similarity that deals with the annealing of hard substrate or solids. The procedure presented in this paper is based on simulating the cooling of substantial matter in a heat tub. This methodology is called annealing.

The law of thermodynamics state that at temperature, t, the probability of an increase in energy of magnitude, δE, is given by (6)

$$\delta E) = e^{-\delta E / kt} \quad (6)$$

Where k = Boltzmann's constant.

Equation (1) is directly utilized in simulated annealing, even though the Boltzmann constant is usually dropped, as this was only introduced into the equation to cope up with and correctly represent different materials. Hence, the probability of accepting a worse state is given by the equation (7) and (8)

$$P = \begin{cases} 1 & if\ \delta c \leq 0 \\ e^{-\delta c / t} & if\ \delta c > 0 \end{cases} \quad (7)$$

$$P = \exp(-c/t) > r \quad (8)$$

Where c = change in the evaluation function
t = current temperature
r = random number lying between 0 and 1

The probability of accepting a worse move is a factor dependent on the temperature of the system as well as the change in the respective cost function. It can be deduced that as the temperature of the system decreases the probability of accepting a worse move also decreases as shown in Figure 2. This is equivalent to gradually moving to a frozen state in realistic physical annealing. Furthermore, if the temperature is zero, only then better-optimized moves will be accepted which effectually makes simulated annealing act like hill climbing (C. S. Jeong1990)

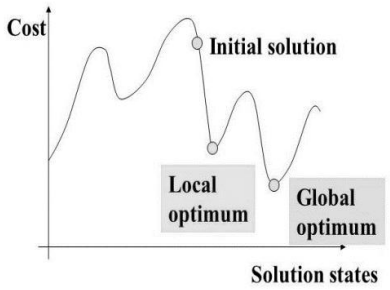

Figure 2: Simulated Annealing with Local and Global optimum

2.4 Proposed Algorithms for leveraging Parallel SA for TSP

The experimentation starts with exploring and examining the feasibility of implementing an execution solver for TSP in MATLAB. In the process of analyzing, it was observed that an operation for exhibiting Simulated Annealing can be programmed and configured; and can be delivered as a probabilistic enhancing method, capable of being used on huge distinct search spaces like the ones undergoing examination in this paper. It works by primarily subjecting the system to situations of lower energy. Continuously, the entity will attain a state where it is difficult to get a lesser drive state which in turn source halts the hunt. These meta-heuristics practice the idea of Temperature which will be connected with new answers. Figure 3 below displays the relative correspondence between the temperature and the new results recycled in the simulated annealing meta-heuristic as described in the execution.

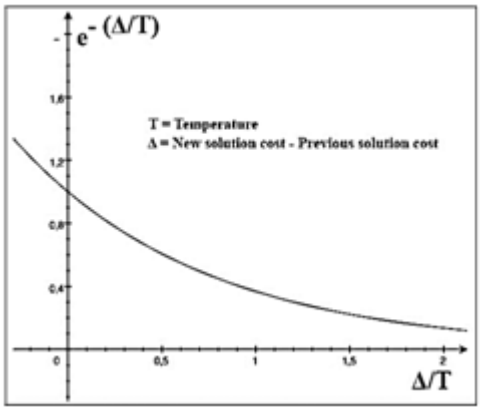

Figure 3: Relation between new solutions with SA and Temperature

The proposed algorithm for the Travelling Salesman Problem using Simulated Annealing is demonstrated below.

Procedure: Simulated Annealing

Begin

1. Find the FIRST tentative route
2. town=randperm(n) //random permutation of the first n integers
3. Tdist=D(town(n), town(1))
4. Tdist=Tdist+D(town(i), town(i+1)) for i=1 till n-1
5. set initial temperature
6. start while loop and stop if no changes for 100 iterations
7. randomly chooses a town (at position c in route)
8. **if** c==1**then**
   previous=n;
   next1=2; next2=3;
   elseif c==n-1 **then**
   previous=n-2;
   next1=n; next2=1;
   elseif c==n **then**
   previous=n-1;
   next1=1; next2=2;
   else previous=c-1; **then**
   next1=c+1; next2=c+2;
9. Do increment in length of the route
10. accept or discard change to route
11. **if** delta<0 or (exp(-delta/T)>= rand)
    swap order of town(c) and town(c+1) in route
    Tdist=Tdist+delta;
    if delta~=0 **then**
    i=0;
    else i=i+1;
    endthe loop
12. set temperature T=0.999*T;

End

In order to achieve parallelization, looping constraints based on the number of processors were additionally incorporated into the algorithm as depicted below.

Procedure: Parallelization
Begin
1. Parallelization of for loop using par for operation
2. Use the simulated annealing algorithm for distance calculation
3. Plot the same using the plot function

End

As part of the Design Algorithm, changes were made to the original parallel simulated annealing algorithm through the insertion of a nil cost path.

Procedure: Design Algorithm with modified Simulated Annealing
Begin
1. **Start** the loop for zero-cost finding
2. **for** i = 1: n
   j = i+ 1;
   if (j == n+1)
   j= 1;

end
3. Distance (i , j) = 0;
4. Distance (j , i) = 0;

End

2.5 Model Architecture and Operations

The solution proposed can be categorized into three main operations:

- tsp2.m: This method creates pseudo-random variables using normal scattering between 0 and 1. The first path is fixated and after that, we can select a new path in case the cost or length is better or lesser than the previous one. This method is constantly applied over a space symmetric matrix also called an adjacency matrix in the software department.
- tsp.m: This function is an extension of the above method but includes the Simulated Annealing algorithm to tackle the problem.
- PARtsp.m: This method makes use of tsp and tsp2 methods defined above and runs the solver with a "parallel" simulation of P processes and n towns.

The "parallel execution" that we refer to, does not imply that various searches need to be made for attaining one global solution in just one simulation. We use PARtsp (ncities, nprocs) function for the exact accurate simulation of n towns which is performed nprocs times, but we make sure that all of them run sequentially at the same time. Each method implements the same program but with changed random numbers and what they get at the end are different costs. Hence executing PARtsp will make us interpret a parallel solution to the problem and result in optimized and lesser-cost solutions for all methods.

There can be a lot of methods to discover a path without cost. Mainly we will make the algorithm discover a path passing all cities where there is no cost related to it. In this specific case, we format a zero-cost path in the humblest way which is conferring to the order of the towns. For instance, consider we take four cities {A, B, C, D}. Then we infer that the distance between cities A and B is zero, the same is the distance between cities B and C, C and D, and at the end also between D and A. We need to recall that past this solution it still remains (4!−1) paths where the total distance is not equivalent to zero.

The alteration was additionally added to the PARtsp.m function right beyond the loop that blocks the distance matrix. This process basically travels the distance matrix through the key diagonal by fixing the cost among the real initial city and the one next to it at zero, in cross commands. An outstanding condition happens when it touches the last city and it must then return to the first one. Once we used a bidirectional graph representation to store the distance between all cities, we easily realize that the original matrix only has the main diagonal with zero values as shown in the Matrix Table1, which matches the distance between a city and itself.

Table 1: Matrix Representing Distances between Cities

D =

0	2.0968	1.7297	4.5634	6.3701	3.1868
2.0968	0	0.3682	2.9365	7.6584	5.2488
1.7297	0.3682	0	3.1640	7.4248	4.8816
4.5634	2.9365	3.1640	0	10.5866	7.2211
6.3701	7.6584	7.4248	10.5866	0	6.2161
3.1868	5.2488	4.8816	7.2211	6.2161	0

Once the zero-cost route is fixed, the distance matrix will currently appear nearly in correspondence to the one shown in Matrix Table 2 below.

Table 2: Matrix Representing Distances between Cities with zero cost Neighbors

D =

0	0	1.7297	4.5634	6.3701	0
0	0	0	2.9365	7.6584	5.2488
1.7297	0	0	0	7.4248	4.8816
4.5634	2.9365	0	0	0	7.2211
6.3701	7.6584	7.4248	0	0	0
0	5.2488	4.8816	7.2211	0	0

2.6 Parameter Configuration

Once we were done with our implementation, we executed a specific number of simulation tests in order to comprehend the performance of the meta-heuristic and reach the main results and conclusions about it. For the simulation, we only create a random x and y only once. We will use the same x and y and run the code for 6, 36, and 216 cities with 4, 8, and 16 processes. We take on this plan so we can confirm if there is a similarity among the results obtained from the program for the various cities and multiple amounts of processes. Here we are going to examine the simulation processing time and overall distance for equating the two solutions attained from Simulated Annealing as shown in Table 1, and another result without Simulated Annealing as shown in Table 2. The given time amount was utilized using the tic and toc functions.

In order to comprehend the performance of the meta-heuristic and reach the main results and conclusions about it, a certain number of tests were performed. For simulation purposes, random x and y were created only once and were used to run the code for 6, 36, and 216 cities with 4, 8, and 16 processes. This was done in order to confirm if there is any similarity among the results were obtained from the program for the various number of cities and multiple amounts of processes. Here we are going to examine the simulation time and overall distance for equating the two solutions attained from Parallel Simulated Annealing as shown in Table 3, and another result with the serial Simulated Annealing algorithm as shown in Table 4. The given time amount was utilized using the tic and toc functions.

Table 3: Execution times with Parallel SA for different number of processes

	Execution Time (sec) (with Parallel SA)		
N Cities	NP=4	NP=8	NP=16
6	1, 697881	2, 359874	5, 884847
36	1, 131469	2, 161275	4, 196453
216	1, 099692	1, 941346	3, 045308

Table 4: Execution times with traditional Serial SA for different numbers of processes

	Execution Time (sec) (with Serial SA)		
N Cities	NP=4	NP=8	NP=16
6	2, 59231	2, 683813	6, 038509
36	1, 185353	2, 685516	4, 976031
216	1, 273476	2, 240703	3, 367285

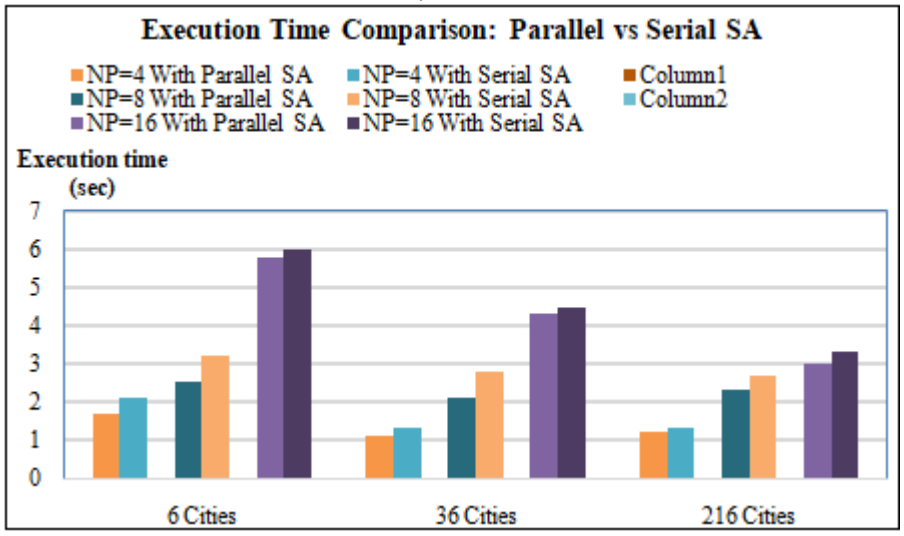

Figure 4: Solution with SA VS without SA for different number of processes

From Figure 4, it is evident that the execution time of the process by means of parallel Simulated Annealing is less as compared to the traditional serial Simulated Annealing algorithm.

3. Simulation Result

As we inferred that the performance analysis above, the execution time taken to find the most optimized path was considerably less for parallel SA as compared to Serial SA. Furthermore, we can make certain deductions about total distances equating the consequences with and without Parallel computed SA in lieu of the modifications we made in the program by injecting a zero-cost route between every pair of cities. The given below images, Figures5, 6, 7 and 8 show the results we get with the x-axis and y-axis both depicting the number of processes and shortest distances for the cities respectively.

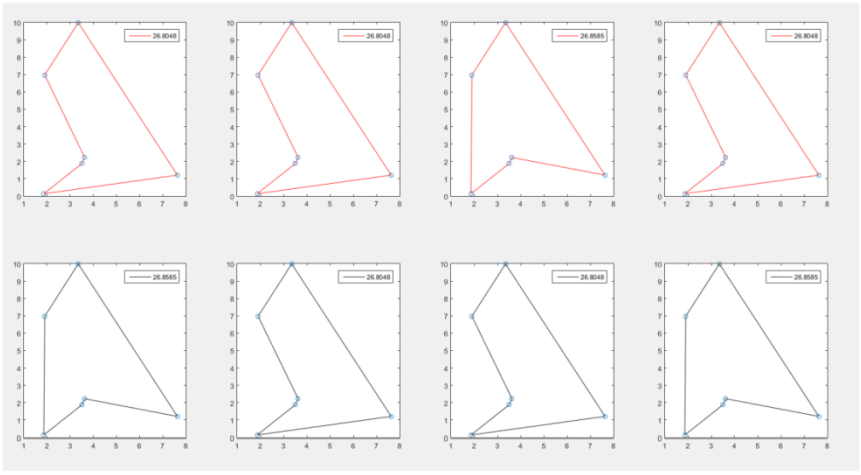

Figure 5: Comparing the results of Parallel SA and Serial SA for 6 cities and 4 processors

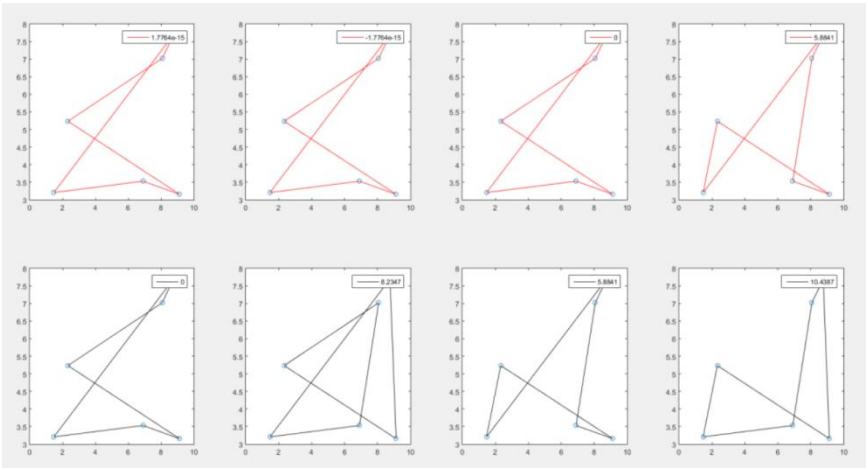

Figure 6: Comparing the results for Zero cost solution of Parallel SA and Serial SA for 6 cities and 4 processors

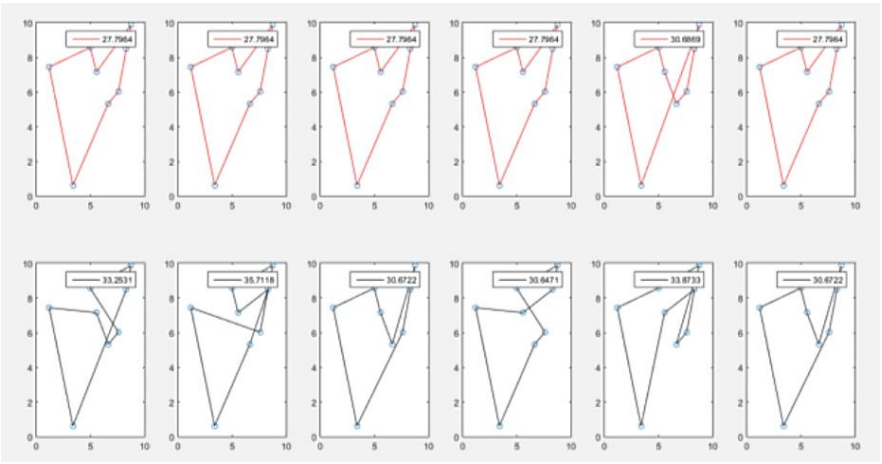

Figure 7: Comparing the results of Parallel SA and Serial SA for 6 cities and 8 processors

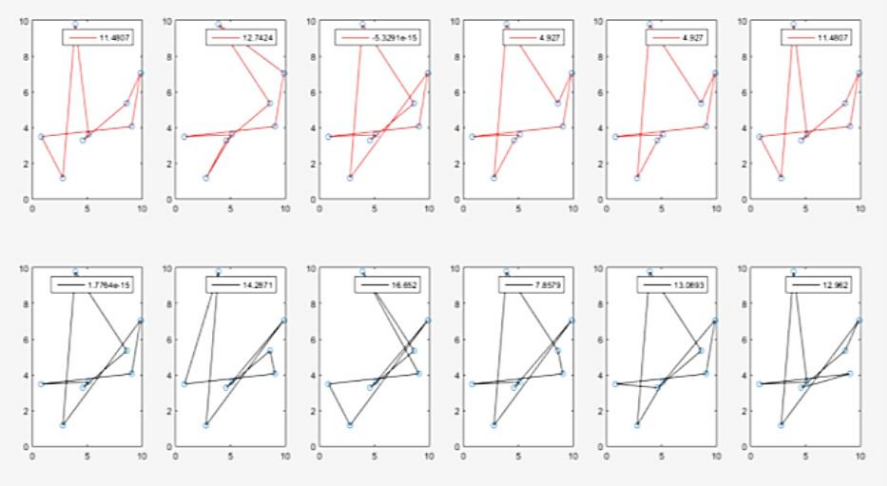

Figure 8: Comparing the results for Zero cost solution with SA and without SA for 6 cities and 8 processor

From Figures5, 6, 7, and 8, we can observe that on running the solver with the parallel Simulated Annealing algorithm, we are able to discover minor distances for a different number of processes. Another point to be considered is that as long as we increase the number of processes, the algorithm seems to get better results related to the shortest path. This in turn validates as well as demonstrates the concept behind simulated annealing; Accepting worse solutions (like the analogy of climbing a mountain) can be advantageous if after overcoming the mountain peak, we achieve a lower solution - in this algorithm, a lower total distance.

4. Conclusion

In this paper, we analyzed the TSP algorithm with a Parallel Simulated Annealing meta-heuristic. We injected a zero cost/distance path between each pair of cities and made a study about the total distances achieved and their performance analysis. This analysis was made for a different number of cities and a different number of processes. We conclude that Parallel Simulated Annealing promises a more optimal solution with reduced execution time, the extent depending on the number of cities and the number of processes chosen. It was also observed that climbing to costly solutions is better if done after overcoming the "mountain peak", as this can lead to a lower global cost - in this case, the lower distance between all cities. To get the minimum cost path we also made some changes to the distance matrix. After the addition of the zero-cost path, the matrix appeared to have a thicker main diagonal with zero values because its adjacent diagonals also became zero. Moreover, the beginning and the last elements of the secondary diagonal also added up to zero as they portray the distance of returning to the first city. After the calculation of the distance matrix, the experimentation was carried out; according to which it was clear that the time spent with Simulated Annealing in parallel distribution was lower and promised a more significantly optimized zero-cost solution when compared to the state-of-the-art algorithm.

5. Conflicts of interest/ Competing interests

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. Hence, the authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] Tsai, H.K., Yang, J.M., Tsai, Y.F. & Kao, C.Y. (2004) An evolutionary algorithm for large traveling salesman problems. IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics, 34, 1718–1729 [DOI: 10.1109/tsmcb.2004.828283] [PubMed: 15462439].

[2] Sumanta Basu & Ghosh, Diptesh, 2008. "A review of the Tabu Search Literature on Traveling Salesman Problems," IIMA Working Papers WP2008-10-01, Indian Institute of Management Ahmedabad, Research and Publication Department.

[3] Rajan, K. Genetically Motivated Search Algorithm for Solving Travelling Salesman Problem (2009).

[4] Kumar & Naveen Karambir and Rajiv Kumar. A Study of Genetic Algorithm to Solve Travelling Salesman Problem Journal of Global Research in Computer Sciences 3 (2012), 33–37.

[5] Singh, A. & Narayan, D. A Survey Paper on Solving Travelling Salesman Problem Using Bee Colony Optimization (2012).

[6] Basu, S. (2012) Tabu search implementation on traveling salesman problem and its variations: A literature survey. American Journal of Operations Research, 02, 163–173 [DOI: 10.4236/ajor.2012.22019].

[7] Hingrajiya, K.H., Gupta, R.K. & Chandel, G.S. (2018) An ant colony optimization algorithm for solving travelling salesman problem. International Journal of Scientific and Research Publications, 2 (ISSN, 2250–3153).

[8] Gupta, Saloni & Panwar, Dr. (2013). Solving Travelling Salesman Problem Using Genetic Algorithm. International Journal of Advanced Research in Computer Science and Software Engineering. 3. 376-380.

[9] Khattar, S. & Goswami, P. (2014) A solution of genetic algorithm for solving traveling salesman problem. [IJSRD - International Journal for Scientific

Research & Development IJSRD] Vol. 2, Issue 04, 2014

[10] Jeong, C.S. & Kim, M.H. Fast parallel simulated annealing for traveling salesman problem (1990). IJCNN International Joint Conference on Neural Networks, 1990, pp. 947–953, vol. 3 [DOI: 10.1109/IJCNN.1990.137955].

[11] Kirkpatrick, Scott & Jr, D. &Vecchi (1983). Optimization by Simulated Annealing Mario. Science, 220, 671–680 [DOI: 10.1142/9789812799371_0035].

[12] Applegate, D.L., Bixby, R.E., Chvatal, V. & Cook, W.J. (2007). The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press: USA.

[13] Björklund, A., Husfeldt, T., Kaski, P. & Koivisto, M. (2012) The traveling salesman problem in bounded degree graphs. ACM Transactions on Algorithms, 8, 1–13, 13 pages, article 18 [DOI: 10.1145/2151171.2151181].

[14] Zhan, S.H., Lin, J., Zhang, Z.J. & Zhong, Y.W. (2016) List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. Computational Intelligence and Neuroscience, 2016, 1712630 [DOI: 10.1155/2016/1712630] [PubMed: 27034650].

[15] Lin, Y., Bian, Zheyong & Liu, X. (2016) Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing tabu search algorithm to solve the symmetrical traveling salesman problem. Applied Soft Computing, 49, 937–952 [DOI: 10.1016/j.asoc.2016.08.036].

[16] Ezugwu, A.E.-S., Adewumi, A.O. &Frîncu, M.E. (2017) Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. Expert Systems with Applications, 77, 189–210 [DOI: 10.1016/j.eswa.2017.01.053].

[17] Behnck, L.P., Doering, D., Pereira, C.E. & Rettberg, A. (2015) A modified simulated annealing algorithm for SUAVs path planning. IFAC-PapersOnLine, 48, 63–68 [DOI: 10.1016/j.ifacol.2015.08.109].

[18] Mijwil, M. (2016). Travelling Salesman Problem Mathematical Description. 10.131 40/RG.2.2.27113.62563.

[19] Garfinkel, R.S. & Gilbert, K.C. (1978) The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. Journal of the ACM, 25, 435–448 [DOI: 10.1145/322077.322086].

[20] Mömke, T. (2015) An improved approximation algorithm for the traveling salesman problem with relaxed triangle inequality. Information Processing Letters, 115, 866–871 [DOI: 10.1016/j.ipl.2015.06.003].

[21] Held, M. & Karp, R.M. (1970) The Traveling-Salesman Problem and Minimum Spanning Trees. Operations Research, 18, 1138–1162 [DOI: 10.1287/opre.18.6.1138].

[22] Ingber, L. (1993) Simulated annealing: Practice versus theory. Mathematical and Computer Modelling, 18, 29–57, ISSN 0895-7177 [DOI: 10.1016/0895-7177(93)90204-C].