**ISSN:**
Print - 2277 - 0593
Online - 2315 - 7461


**Journal of Natural Sciences, Engineering and Technology**

# SIMULATED ANNEALING FOR SOLVING THE TRAVELLING SALESMAN PROBLEM

**[1]*A.P. ADEWOLE, [2]S.O.N. AGWUEGBO AND [1]K. OTUBAMOWO**

[1]Department of Computer Science,
[2]Department of Statistics,
Federal University of Agriculture, Abeokuta, Nigeria.
*__Corresponding author__: philipwole@yahoo.com **Tel:** +2348034404207

## ABSTRACT

In this paper we considered the use of Simulated Annealing for solving the Travelling Salesman Problem (TSP) which is NP-Complete. The algorithm searches solutions for the global minimum by perturbing existing solutions and replace if the new solution is better than the existing solution. The annealing process is controlled by some algorithmic parameters and thus the solution relies on the parameters set. Different parameters were used to know the best set of parameters. Generally the algorithm was tested and proved to be a good solver of TSP.

**Keywords**: Global minimum, Local Minimal, NP-complete problem, Optimal Solutions, *Simulated Annealing,* Traveling Salesman Problem

## INTRODUCTION

The traveling salesman problem (TSP) is a well-known and important combinatorial optimization problem. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. In contrast to its simple definition, solving TSP is difficult since it is an NP-complete problem. Apart from its theoretical appeal the TSP has many applications. Typical applications in operations research include vehicle routing, computer wiring, cutting wallpaper and job sequencing. The main application in statistics is combinatorial data analysis, e.g., reordering rows and columns of data matrices or identifying clusters. The NP-completeness of the TSP already makes it more time efficient for small-to-medium size TSP instances to rely on heuristics in case a good but not necessarily optimal solution is sufficient.

In this paper, Simulated Annealing algorithm is used to solve Travelling salesman Problem. Simulated Annealing is a probabilistic method used for finding the global minimum of a cost function that may possess several local minimal (Bertsimas and Tsitsiklis). It is a Monte Carlo maximization/minimization technique used for complex problems with many parameters and constraints. It mimics the process of annealing, which starts with a high temperature mix of metal and slowly cools the mix, allowing optimal structures to form as the material cools. The Simulated Annealing procedure randomly generates a large number of possible solutions, keeping both good and bad solutions. These are randomly perturbed and can replace the existing

solutions, even when they are worse. As the simulation progresses, the requirements for replacing an existing solution or staying in the pool becomes stricter and stricter, mimicking the slow cooling of metallic annealing. Eventually, the process yields a small set of optimal solutions. Simulated Annealing's advantage over other methods is the ability to obviate being trapped in local minima. This means that the algorithm does not always reject changes that decrease the objective function or changes that increase the objective function according to its probability function:

$P = \exp(-\Delta f/T)$

Where T is the control parameter (analogy to temperature) and $\Delta f$ is the variation in the objective function. The probability function is definitely a derivative of the Boltzmann probability distribution function (Ali Hamdar 2008).

### *Travelling Salesman Problem*

The TSP is probably the most widely studied combinatorial optimization problem because it is a conceptually simple problem but hard to solve, it is an NP complete problem, A Classical Traveling Salesman Problem (TSP) can be defined as a problem where starting from a node it is required to visit every other node only once in a way that the total distance covered is minimized. This can be mathematically stated as follows:

$$\text{Min} \quad \sum_{i,j} c_{ij} x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{j} x_{ij} = 1 \; \forall \; i \neq j \tag{2}$$

$$\sum_{i} x_{ij} = 1 \forall \quad j \neq i \tag{3}$$

$$u_i = 1 \tag{4}$$

$$2 \leq U_i \leq n \quad \forall' \neq I \tag{5}$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) \tag{6}$$

$$\forall \; i \neq j \forall j \neq 1 \;, \quad u_i \geq 0 \; \forall \; i \tag{7}$$

$$x_{ij} \in \{0,1\} \forall i,j \tag{8}$$

Constraints set (4), (5), (6) and (7) are used to eliminate any sub tour in the solution. Without the additional constraints for sub tour elimination, the problem reduces to a simple assignment problem, which can be solved as an L.P. (Linear Programming) without binary constraints on $x_{ij}$ and will still result in binary solution for $x_{ij}$. Introduction of additional constraints for sub tour elimination, however, makes the problem an M.I.P. (Mixed Integer Problem) with $n^2$ integer variables for a problem of size n, which may become very difficult to solve for a moderate size of problem (Sachin Jayaswal 2004).

### *Applying Simulated Annealing To Solving Tsp*

The method is inspired by experimental observations on crystallization from a melt. At high temperatures, the atoms in the melt are free to move around the sample. As the temperature is reduced, the atoms tend to crystallize into a solid. If the sample is quenched, i.e., cooled very rapidly, then the solid is usually polycrystalline or amorphous in form. If the sample is annealed, i.e., cooled slowly, then the sample stands a better chance of forming a perfect crystal, which is the global minimum energy configuration of the system. Defects cost energy, and samples with defects correspond to local minima of the energy. Since there are an enormous number of possible configurations with defects, the energy landscape of the cooled solid is very complicated with numerous hills and valleys. Quenching typically leads to the bottom of the nearest valley, which annealing allows the system to explore the landscape and settle down into one of the lower valleys. The key to successful annealing is to use a good annealing schedule, i.e., a protocol for gradually reducing the temperature of the sample.

## METHODOLOGY

Simulated annealing in statistical physics is based on the concept of thermal equilibrium at temperature T. The probability of a configuration of the system is determined by the Boltzmann factor

$$prob(E) \sim \exp\left(\frac{E}{k_b T}\right)$$

Where E is the energy of the configuration. At T = 0 the system is stuck at a minimum. At finite T, it can move around from one configuration to another, i.e., it can explore the nearby hills and valleys on the energy landscape. To implement simulated annealing, we need a dynamics for the system to move from one configuration to another at a particular temperature T. A natural algorithm to use is the Metropolis algorithm: a random trial step is taken; if the step is downhill on the energy landscape (i.e., if the probability increases) the step is always accepted; and if the step is uphill, it is accepted conditionally. It can be shown theoretically that the minimum energy configuration for this system is a triangular lattice the closest packed configuration of disks in 2-D. (A square lattice is actually unstable and corresponds to a saddle point of the energy.) Local minima correspond to triangular lattices with defects.

Initial Configuration: Either a square lattice or a random initial configuration of atoms. Units are chosen so that the equilibrium separation in the Lennard-Jones potential the average interparticle spacing should of course be $\sim 1$ to ensure that the system is close to being a solid.

Boundary Conditions: Periodic boundary conditions can be used to model a bulk system. Free boundary conditions are more appropriate to model clusters of atoms.
Effective Temperature: Silverman and Adler found that using an effective temperature for each atom yielded good results. They argue that when the particle makes a trial move, the change in energy $\Delta$ E will be roughly proportional to the number of neighbors, so the acceptance probability

$$\exp\left(-\frac{\Delta E}{zk_BT}\right)$$

will be less dependent on the number of neighbors.

Recall that one Monte Carlo step per particle is conventionally taken to be N Metropolis steps with the particles being chosen either at random or in sequence. An annealing schedule is a sequence of temperatures Ti and Monte Carlo steps per particle $n_i$ , $i$ = 0,1, 2,. . .. There is evidently a lot of freedom in choosing a schedule.

### *Algorithm*

1. Initialization: Generate a random candidate route and calculate fitness value for the route.
2. Repeat following steps NO OF ITERATIONS times:
(a) Neighbor function: Generate a neighbor route by exchanging a pair of cities.
(b) Acceptance function: Evaluate the neighbor route for acceptance - if accepted, replace current route with neighbor route.
3. Return the best result.

## IMPLEMENTATION

### *Program Details*

The program was written with java. A TSP class was created which has 4 methods and 15 instance variables. The methods and their functions are explained below.

**OpenFile():** This method initializes currentOrder and nextOrder and then displays a JFileChooser that lets you browse for the text folder that contains the distances matrix. It also reads the file and sets disances[i][j] according to the text folder.

**GetTotalDistance ():** Calculates the cost of a tour which is input as a parameter in form of a list.
**GetTotalDistance ():** Calculates the cost of a tour which is input as a pameter in form of a list.
**GetNextArrangement ():** Finds a new tour that does not exist previously by randomization
**Anneal ():** The iterations are carried out here, new tours are continuously generated based on previous tours until no more visible exist in the new tours created.
In the interface the button load triggers the method openFile and also causes the distances matrix to be displayed on the JTextArea labeled input. The other button on the form which is the solve button calls the method Anneal() which solves the problem loaded previously and displays the result on the second JTextArea labeled output.

In the program there are three algorithmic parameters that can be altered at each run of the program so as to vary the annealing schedule. The three parameters are

- Absolute Temperature
- Cooling Rate
- Initial Temperature

The parameters go long way in determining the result of the algorithm. The program generates a random tour using Nearest

neighbor algorithm. The nearest neighbor algorithm (Rosenkrantz etal. 1977) follows a very simple greedy procedure: The algorithm starts with a tour containing a randomly chosen city and then always adds to the last city in the tour the nearest not yet visited city. The algorithm stops when all cities are on the tour. An extension to this algorithm is to repeat it with each city as the starting point and then return the best tour found. This heuristic is called repetitive nearest neighbor. Whenever a new neighbor tour is generated the temperature is cooled by a percentage specified by the cooling rate. This procedure continues until the absolute temperature is reached. The Initial temperature is chosen based on the size of the problem. The cooling rate should be a number that is very close to one so that the temperature can be cooled gradually until it reaches the absolute temperature.

To test the algorithm a Geometric city with 7 nodes is used. The optimal tour of the geometric city is 6->5->4->3->2->1->0 or 0->1->2->3->4->5->6, both have the same cost.

Simulated Annealing algorithm was tested and the result is shown on the screen capture below.

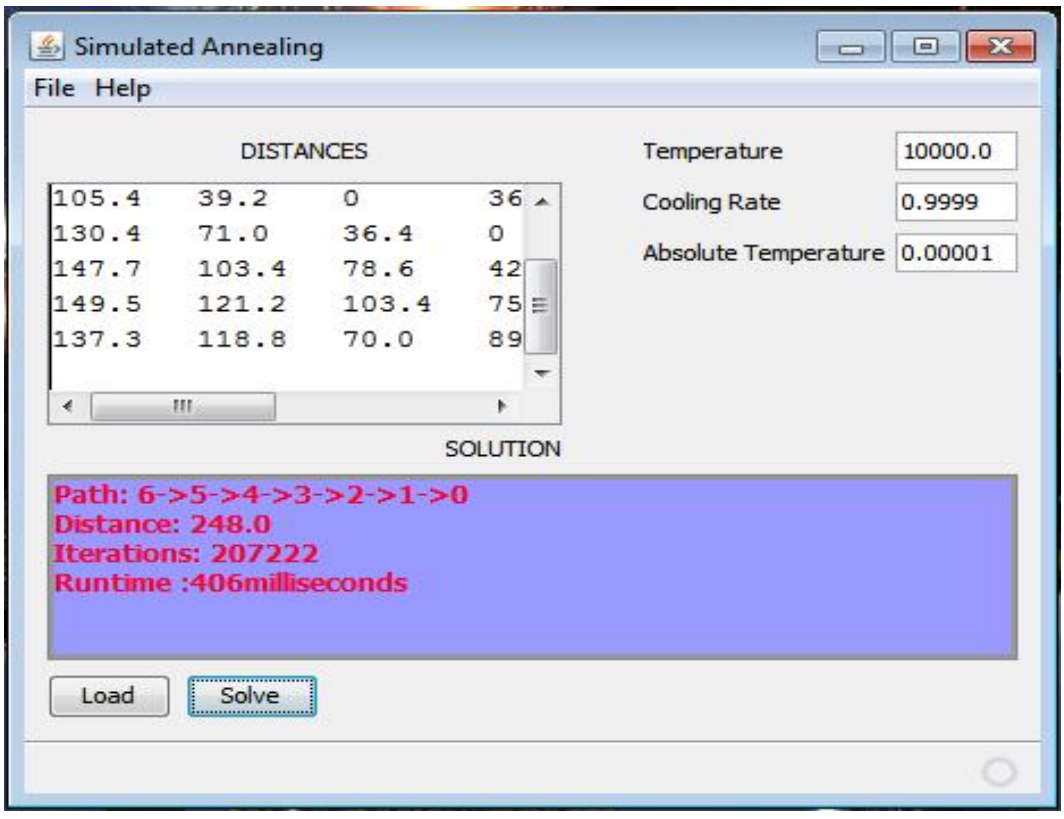


**Fig. 1: Using the Algorithm to solve a Geometric City**

The program is used to solve a geometric city with clear optimal solution so as to be sure that the algorithm can arrive at optimal solution. As we can see in the picture above the path is optimal and the run time is fair at 406 milliseconds.

## RESULTS AND DISCUSSIONS

The data used in this paper is the distances between Nigerian major cities. This data was used because it is an average sized problem with 31 cities and its distances are moderate. The data is stored in a text file which can be imported into the program by clicking the "load" button on the GUI.

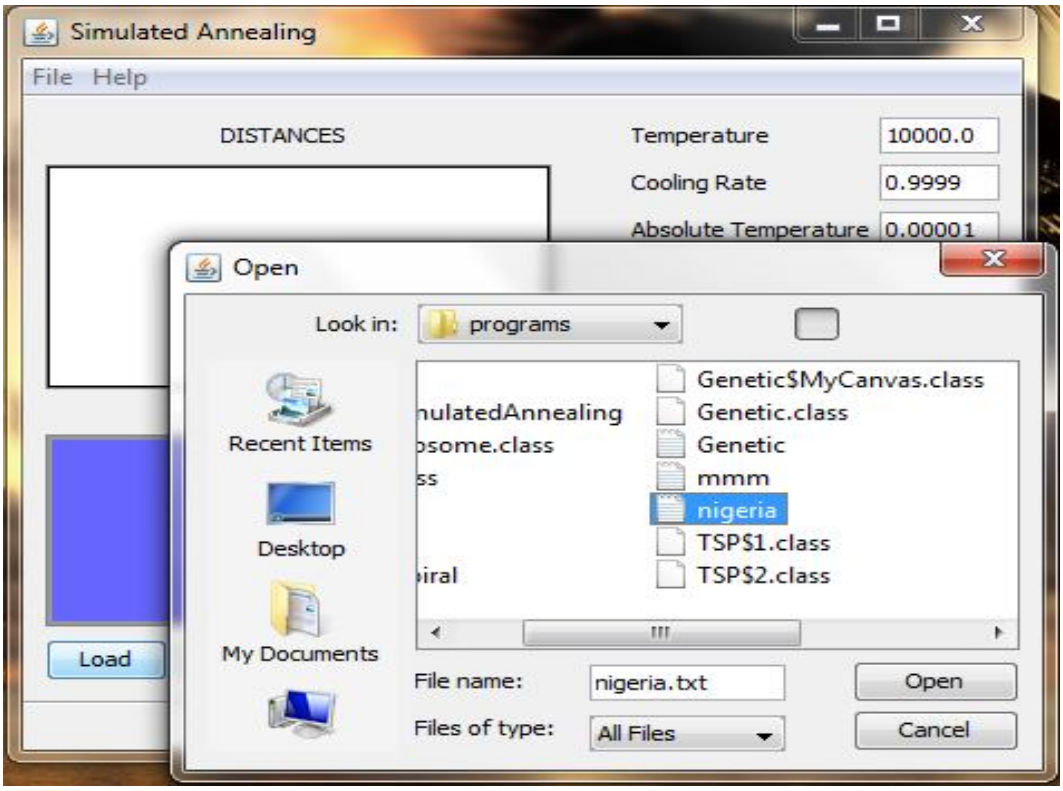


**Fig. 2: Loading cities from text file**

The Table below shows the results of experiments carried out on the algorithm using different parameters. The table shows all the parameters used and the results. Performance of the result is based in the run-time and Distance (cost).

**Table 1: Parameters and Results**

| Temp | Cooling Rate | Abs Temp | Run Time | Distance | Iterations |
|---|---|---|---|---|---|
| 10000 | 0.9999 | 0.0001 | 921 | 6239 .0 | 184197 |
| 10000 | 0.999 | 0.0001 | 109 | 7489.0 | 18411 |
| 10000 | 0.99 | 0.0001 | 0 | 8211.0 | 1832 |
| 10000 | 0.9 | 0.0001 | 0 | 12795.0 | 174 |
| 1000 | 0.9999 | 0.0001 | 780 | 6368.0 | 161172 |
| 1000 | 0.999 | 0.0001 | 78 | 7033.0 | 16110 |
| 1000 | 0.99 | 0.0001 | 0 | 7495.0 | 1603 |
| 1000 | 0.9 | 0.0001 | 0 | 7858.0 | 152 |
| 100 | 0.9999 | 0.0001 | 592 | 5163.0 | 138148 |
| 100 | 0.999 | 0.0001 | 62 | 5700.0 | 13808 |
| 100 | 0.99 | 0.0001 | 0 | 5669.0 | 1374 |
| 100 | 0.9 | 0.0001 | 0 | 5872.0 | 131 |

From the results obtained it was found that increase in temperature leads to increased run time. The runtime is also affected by the rate of cooling. The rate of cooling determines the no of iterations. The graph below shows the plot of runtime against cooling rate.

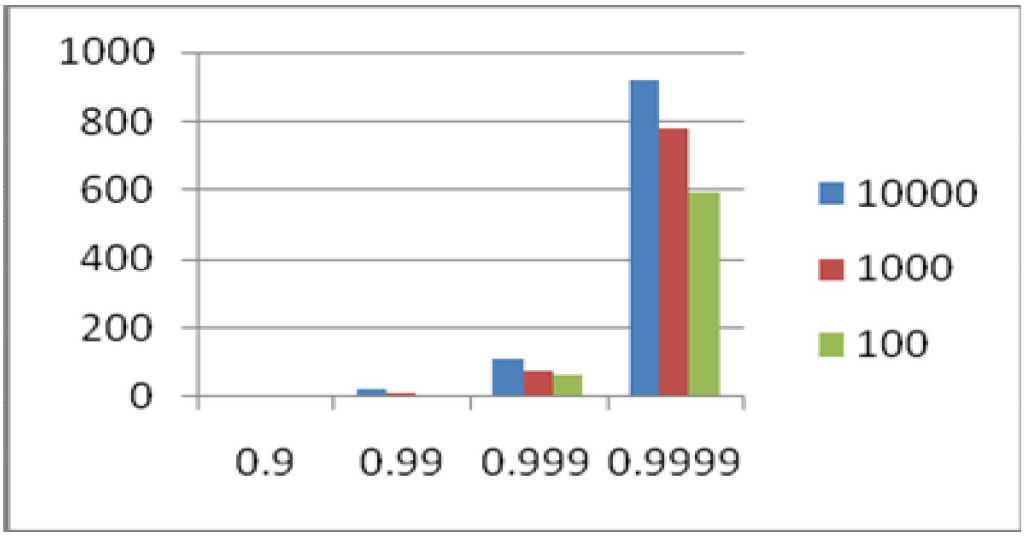


**Fig. 3: Plot of runtime against cooling rate**

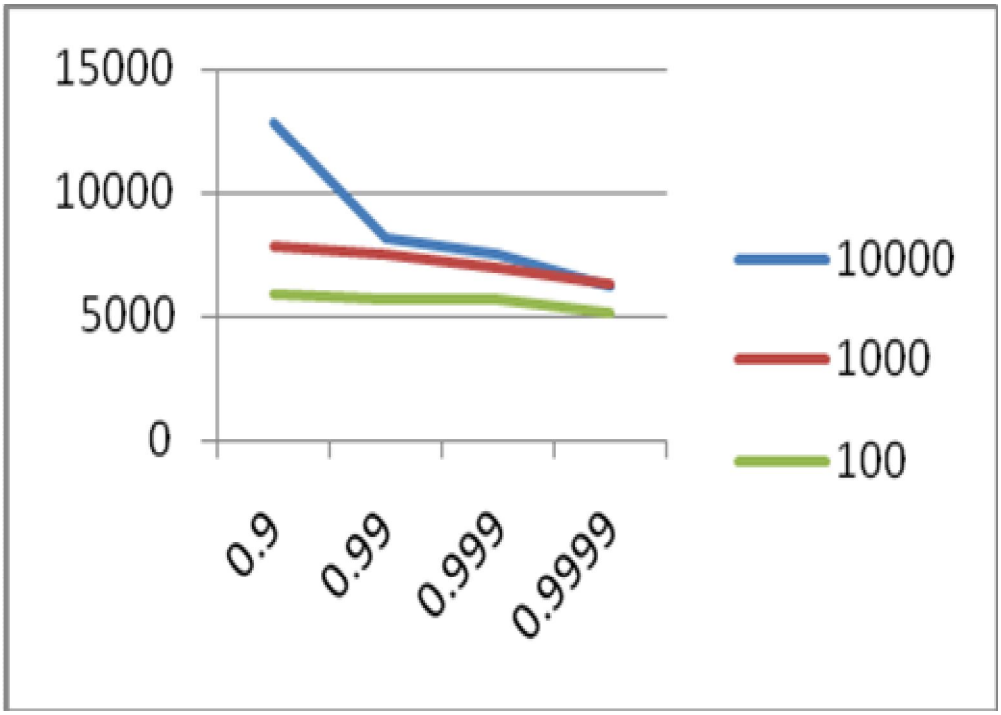


**Fig. 4: Plot of Distances against cooling**

From the graph above we can infer that the cooling must be very close to one so as to get better solutions. The Initial temperature should be set according to the problem size because it can either be too large or too small for the problem. However, this should be considered if the runtime is being considered. The absolute temperature should be set to be as low as possible depending on the quality of solution we require. Combination of parameters that provides a good balance of runtime and solution quality for medium size problems is given below.

**Table 2: Parameters with best performance**

| Parameter | Value |
|---|---|
| Temperature | 1000 |
| Cooling Rate | 0.9999 |
| Absolute Temperature | 0.0001 |

This combination has been tested with various problem sizes less than 100 and it has produced best results on the average compared to other combinations.

## CONCLUSION

Simulated Annealing is a very good solver of the Travelling Salesman Problem. The annealing schedule (parameters) is a very crucial factor in the performance of the algorithm. The performance is measured on the scale of runtime and solution quality. The parameters used should also be chosen with the problem type in consideration.

## REFERENCES

**Ali, Hamdar** 2008. Simulated Annealing-Solving the Travelling Salesman Problem,*www.codeproect.com*.

**Anthony Ralston, Edwin D. Reilly** 1993. Encyclopedia of Computer Science, *Chapman and Hall*.

**Korte, B.** 1988. Applications of combinatorial optimization, *talk at the 13th International Mathematical Programming Symposium, Tokyo*.

**Bertsimas, D., Tsitsiklis J.** 1993. Simulated Annealing, *Journal of Statistical Science*, 8 (1): 10-15.

**Rosenkrantz, D.J., Stearns, R.E., Philip, Lewis M.** 1977. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3): 563-581.

**Lawler, E.L., Lenstra, J.K., RinnooyKan, A.H.G., Shmoys, D.B.** 1985. The Traveling Salesman Problem, *John Wiley & Sons, Chichester*.

**SachinJayaswal,** 2004. A comparative study of Tabu Search and Simulated Annealing for Travelling Salesman problem.

*(Manuscript received: 29th December, 2009; accepted: 22nd December, 2010).*