Solving Travelling Salesperson Problem using Simulated Annealing

Jagadeesh Prasad Batchu
Department of Computer Science
University of Exeter
Exeter, UK
jb1473@exeter.ac.uk

***Abstract*—The Travelling Salesman Problem (TSP) is a classic optimization that has applications in various fields, including logistics, transportation, planning, microchips, DNA sequencing and computer science. The TSP involves finding the shortest possible route that a salesman can take to visit all the cities in a given set, returning to the starting city[1]. Simulated annealing is a metaheuristic optimization method[2] that has been successfully applied for solving unconstrained and bound-constrained optimization problems[3] such as TSP. In this paper, we explore the use of simulated annealing for solving the TSP and compare it with other search algorithms.**

I. Introduction

The Travelling Salesman Problem is one of the most extensively studied combinatorial optimization problems in the field of computer science which is a problem first formulated in 1930. The problem involves finding the shortest possible path that a salesman can take to visit all the cities in a given set, returning to the starting city[1]. The TSP is a challenging problem because it is NP-hardness (non-deterministic polynomial-time hardness), meaning that there is no known algorithm that can solve it in polynomial time in combinatorial optimization[1]](a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects)[4]. Therefore, heuristic algorithms such as simulated annealing have been used to find approximate solutions to the TSP.

Simulated annealing is a stochastic optimization technique that is inspired by the physical process of annealing which is a technique used in Mechanical Engineering(Metallurgy)[5]. In simulated annealing, a random initial solution is generated and iteratively improved by making small random changes to the solution. These changes are accepted or rejected based on a probability distribution that depends on the current temperature of the system. As the temperature decreases, the acceptance probability becomes more stringent, and the algorithm converges toward a local minimum[3].

Simulated annealing has been applied to the TSP with promising results. The algorithm has been shown to produce high-quality solutions with a reasonable running time, even for large instances of the TSP. In this paper, we will discuss the application of simulated annealing to the TSP and compare its performance to other heuristic algorithms. We will also examine the impact of various parameters on the performance of simulated annealing and discuss the limitations of the approach.

When Simulated annealing was compared to the genetic algorithm for solving the TSP. The study found that simulated annealing produced the best results in terms of solution quality and running time[6].

II. BACKGROUND

A. Traveling Salesman Problems

TSP(Traveling Salesman Problems) is an NP-hard problem where no polynomial time algorithm is known to guarantee its global optimal solution[7]. The Traveling Salesman Problem has to find a short path that it can travel through all cities without visiting the same city twice and at the end of the journey the salesperson has to return to the starting city where the journey began.

The problem data can be defined in the form of a symmetric distance matrix, asymmetric distance matrix, and coordinates. In a symmetric distance matrix, the distance between two distinctive cities is the same among all cities(A to B == B to A) and it is represented in n x n matrix where n is a number of cities and all diagonal elements in the matrix is 0 and values are in symmetric, In asymmetric distance matrix the distance from city A to city B is not equal to city B to city A(A to B != B to A) where the diagonal is 0 and values are not symmetric, and In coordinates data, the coordinates of the cities are given in the form of (x, y) for each city. In this paper, the symmetric matrix is based on co-ordinates based data is used.

To solve a TSP, theoretically, we can do using permutations but it takes a lot of time to evaluate. For example, to find the shortest distance between 15 cities it will take 15! Calculations which is 1.3 trillion. For n number of cities, it has to calculate n! permutations to get the result.

B. Simulated Annealing

SA(Simulated Annealing) algorithm is the earliest among the metaheuristics and one of the few algorithms that have explicit strategies to avoid local minima to reach global optima. The origins of SA are in statistical mechanics and it was first presented for combinatorial optimization problems. The fundamental idea is to accept moves resulting in solutions of worse quality than the current solution in order to escape from local minima. The probability of accepting such a move is decreased during the search through parameter temperature[7].

$$P = 1, iff(new) < f(x)$$

$$P = e^{-(f(new)-f(x))/t}, otherwise$$

Where f(new) = total distance for the new function generated using mutation(in this case) and f(x) = total distance for the current best function, P = Probability of accepting the new function if the new function has the best value then the probability of acceptance of the new function is 1, And if the new function value is not the best compared to the existing function then it will evaluate the probability function using the formula given above and based on the probability it will decide to accept or reject the new function and there will be a copy of best solution. Here when the number of iterations is increased acceptance of the worst solution decreases.

C. Random Search

The Random Search algorithm is a simple and effective optimization technique used to search for the optimal solution in a large search space without any prior knowledge of the problem domain. It involves randomly selecting candidate solutions from the search space and evaluating their performance against a given objective function[8].

The algorithm starts by generating a random solution and then evaluates its fitness value in Travelling Salesman Problem. Here one best-performing fitness function is stored and compared with the newly generated value. The algorithm then selects the best solution between the two and continues the process for a specified number of iterations or until the desired level of performance is achieved.

Random search is a powerful algorithm that has been widely used in various applications, including machine learning, optimization, and simulation. It is known to be computationally efficient and robust, making it suitable for solving complex problems with large search spaces.

D. Stochastic Hill Climbing

Stochastic Hill Climbing is also a type of local search algorithm that is commonly used to solve optimization problems. The algorithm works by iteratively making adjustments to the solution and accepting any changes that result in an improvement in the objective function, while also allowing for some changes that may result in a worse solution in order to escape local optima[9].

At each iteration, the algorithm generates a new solution and compares its objective function value to that of the current best solution. If the new solution has a better objective function value, it is accepted as the new best solution. If the neighboring solution has a worse objective function value, it may still be accepted with a certain condition such as an acceptable difference between the solutions(new solution – best solution) for Travelling Salesman Problem. This allows the algorithm to explore the search space more broadly, rather than getting stuck in local optima.

III. Experimentation

The experimentation of the Travelling salesman Problem using the Simulated Annealing algorithm, classic method, and other local search algorithms such as Random Search and Stochastic hill climbing is done in Jupyter Notebook using and Python Programming language using several libraries such as NumPy, matplotlib, random, time.

A. Importing Python libraries

Numpy library is a popular Python library used for scientific computing and data analysis. It provides a powerful N-dimensional array object, which can be used for efficient manipulation of large arrays and matrices of numerical data. The Random library is a standard Python library that provides functions for generating random numbers, sequences, and shuffling sequences. From Matplotlib library a sublibrary pyplot, Matplotlib is a popular data visualization library for Python. It provides a range of tools for creating static, animated, and interactive visualizations in Python. Pyplot is a sublibrary of Matplotlib that provides a collection of functions for creating common types of plots, such as line plots, scatter plots, histograms, and bar charts. Time module, this module provides various functions to work with time-related operations such as getting the current time, measuring the time taken to execute a code block, pausing the program execution for a certain amount of time, and many more.

B. Defining the required functions

Functions to calculate Euclidean distance between the cities and the total distance of the given set of generated functions are calculated with the help of these functions. Also, a function to generate a new solution using mutation is written.

C. Classic method

The classic method means the textbook method. Where the salesperson starts in a certain city and travels to the nearest city possible without vising the cities twice. Here the salesperson always starts from the first city/home city and ends at the home city. It provides a simple solution but not an accurate solution. This algorithm function does not iterate several times compared to other optimization algorithms, it will generate one single solution at the end.

Classic Method Algorithm:

1) Select/Start from home city
2) Search the city with the least distance, add them to the function and add the distance.
3) Loop though all remaining cities
    a) Search for least distance city
    b) If city already exists repeat step 3
    c) Add city to function and add distances
4) Add the distance between last city and the home city to the total distance.

D. Simulated Annealing Algorithm.

In the simulated annealing Algorithm, the function takes a randomly generated initial solution, initial temperature, cooling rate, and number of iterations as the input. Here deciding to accept the new solution plays an important role which also depends on the number of iterations and the temperature.

Simulated Annealing Algorithm:

1) Initialize/generate a random solution.
2) Calculate distance.
3) Iterate till it reaches the number.
    a) Generate a new solution using mutation.
    b) Calculate the distance.
    c) Condition I: if new distance $<$ current distance.
        i) Accept and update the solution.
        ii) Go to step 3a.
    d) Condition II : exponential(-(current distance- new distance)/temperature) $<$ Random Probability.
        i) Accept the solution.
        ii) Go to step 3a.
    e) Update temperature.
4) Return Solution and Distance.

In condition 3(d) it calculates the exponential value of the difference between the new solution distance with the current best solution distance which is divided by the current temperature. The temperature value changes along with the increase of the iterations. Here the cooling rate is multiplied by the cooling rate. After several experiments, the effective cooling rate for good results is the division of the number of total iterations subtracted from one and the total number of iterations.

From these conditions, 3a, and 3b the algorithm either accepts the best solution(3a) or the worst solution if it satisfies the condition. And the best solutions are only updated if 3a is satisfied.

Cooling rate = (Iterations -1)/Iterations

E. Random Search Algorithm.

This algorithm randomly generates a solution and compares it with the best solution. This function runs for a certain number of iterations based on the input.

Random Search Algorithm.

1) Initialize distance to max.
2) Iterate till it reaches the number.
    a) Generate a random solution.
    b) Calculate the distance.
    c) Condition: if new distance ¡ current best distance.
        i) Accept the solution and update the best solution.
        ii) Go to step 3a.
3) Return distance and solution.

F. Stochastic Hill Climbing Algorithm.

In this algorithm, it takes a randomly generated solution and a number of iterations as input. The function runs several times based on the iteration number. Each time the function generates a new solution by mutating the current solution and then the new solutions are evaluated

Stochastic Hill Climbing Algorithm.

1) Initialize random solution.
2) Calculate distance.
3) Iterate till it reaches the number.
    a) Generate a new solution using mutation.
    b) Calculate the new distance.
    c) Condition 1: if new distance ¡ current best distance.
        i) Accept the new solution and update the best distance.
        ii) Go to step 3a.
    d) Condition II: if (new distance – current best distance) ¡ = acceptable difference.
        i) Accept the new solution.
        ii) Go to step 3a.
4) Return the best solution and distance.

In this algorithm, the acceptable difference has to be specified manually or logically. Based on that difference it will accept the solution and iterates.

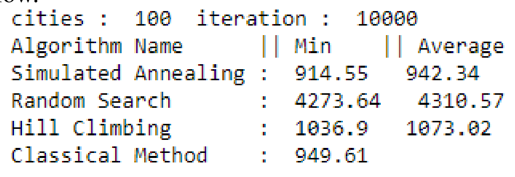
G. Graphs and variables

The functions for plotting the graphs are done with the help of pyplot sublibrary from Matplotlib for an effective visual representation of the results. And the global variables are also initialized such as the number of cities and randomly generated coordinates. Which are used in the algorithm functions.

IV. Results

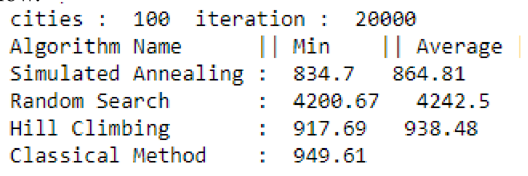
In the Simulated Annealing Algorithm, with the increase in the number of iterations, the value of temperature decreases which results in a decrease in accepting the worst solutions. Where the probability of accepting the worst solutions is higher at the beginning of the iterations.

When the number of cities is given as 100 and the number of iterations is given as 10,000 the minimum distance and the average distance of these algorithms over 5 results are shown below.



```
cities :  100  iteration :  10000
Algorithm Name      || Min    || Average
Simulated Annealing :  914.55   942.34
Random Search       :  4273.64   4310.57
Hill Climbing       :  1036.9   1073.02
Classical Method    :  949.61
```

When the number of cities is given as 100 and the number of iterations is given as 20,000 the minimum distance and the average distance of these algorithms over 5 results are shown below.



```
cities :  100  iteration :  20000
Algorithm Name      || Min     || Average
Simulated Annealing :  834.7    864.81
Random Search       :  4200.67   4242.5
Hill Climbing       :  917.69   938.48
Classical Method    :  949.61
```

A. Classic Method

```
Classic Method :  949.61 | Time  :  0.22
```

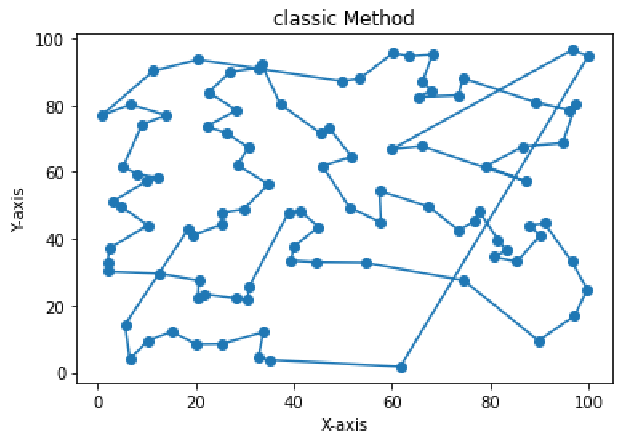


Fig 1: Route for 100 cities using Classic Method

B. Simulated Annealing

```
Simulated annealing :  890.64 | Time  :  1.62
```

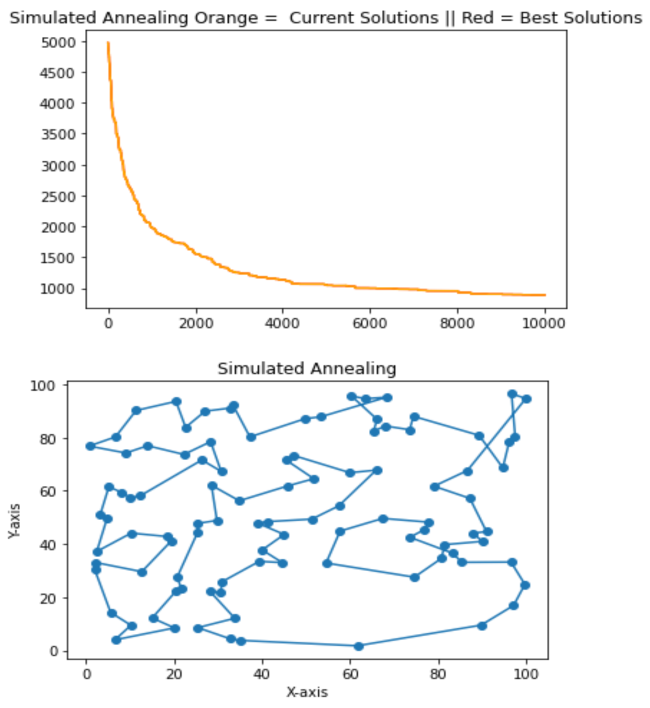


Fig 2(a): Route for 100 cities using Simulated Annealing for 10,000 iterations

```
Simulated annealing :  875.47 | Time  :  3.13
```

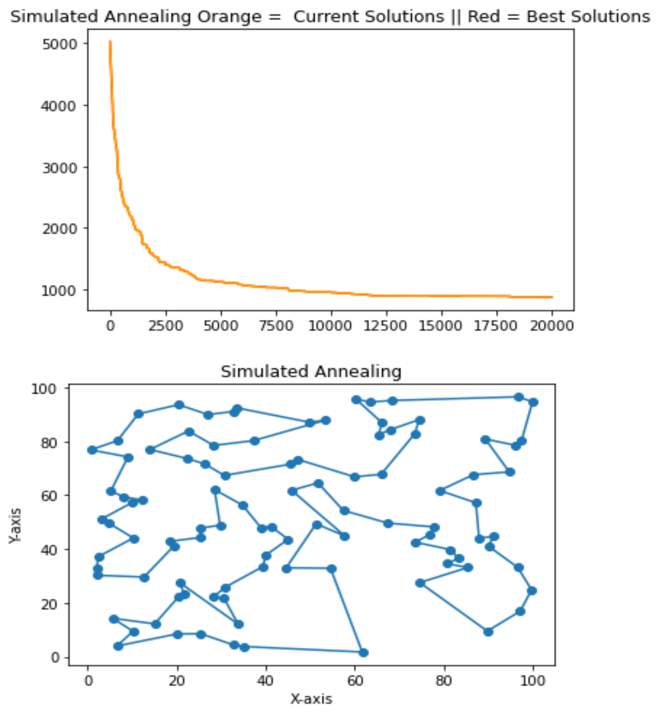


Fig 2(b): Route for 100 cities using Simulated Annealing for 20,000 iterations

C. Random Search

```
Random search distance :  4382.44 | Time  :  1.27
```

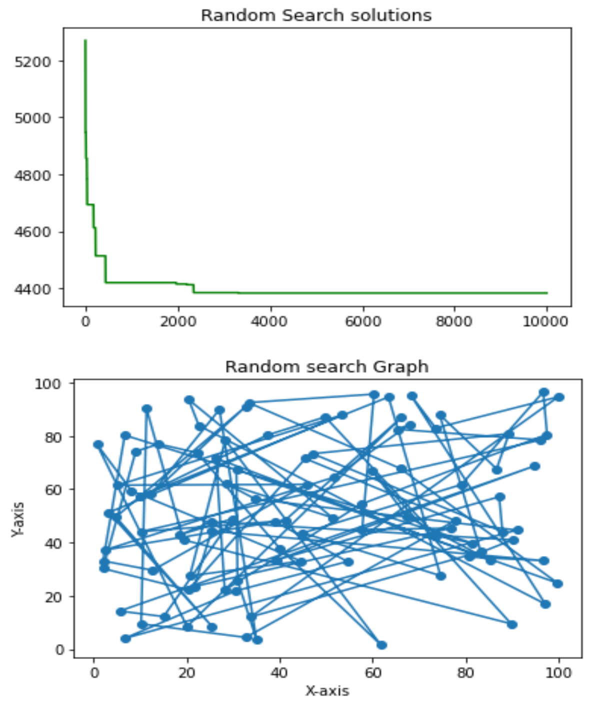
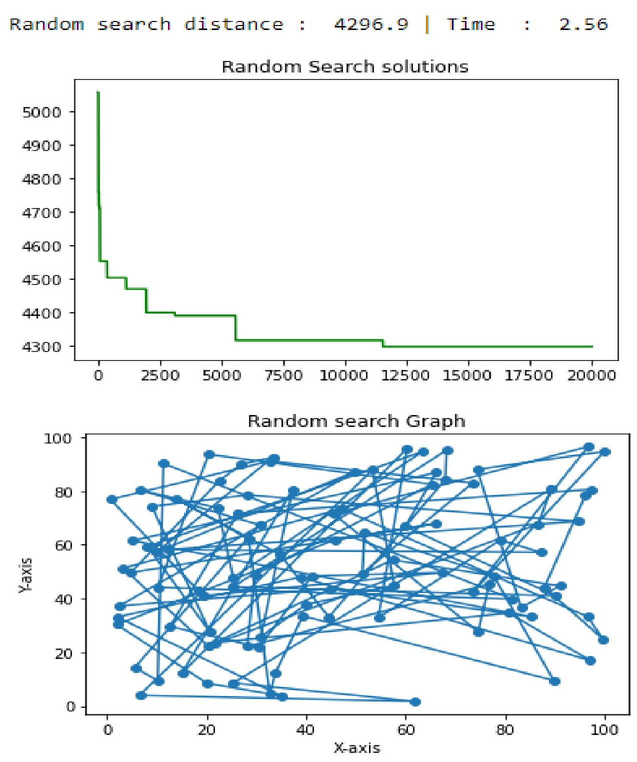


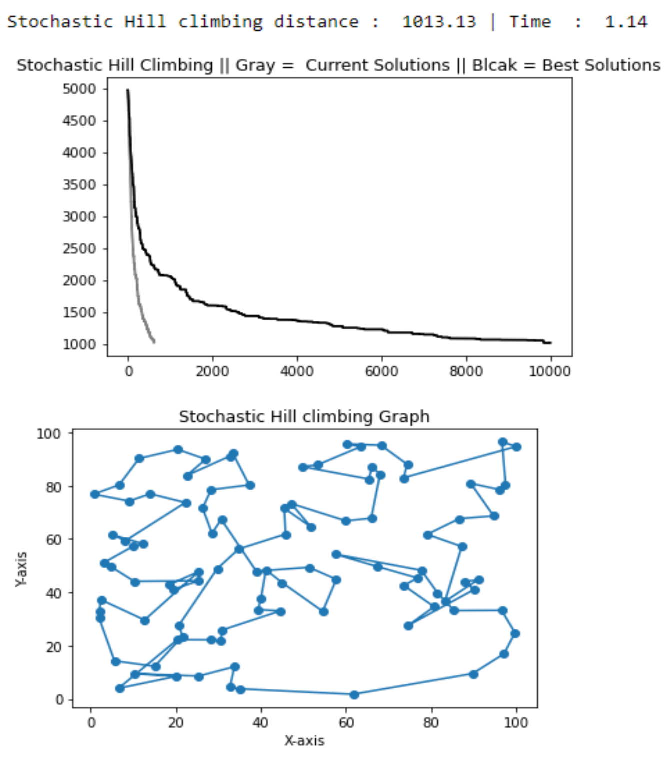
Fig 3(a): Route for 100 cities using Random Search for 10,000 iterations



```
Random search distance :  4296.9 | Time  :  2.56
```

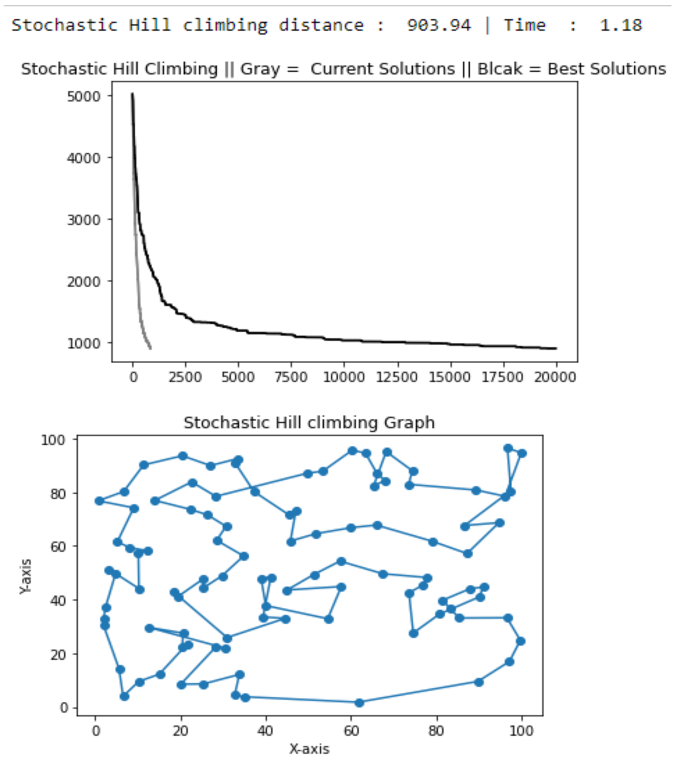
Fig 3(b): Route for 100 cities using Random Search for 20,000 iterations

D. *Stochastic hill Climbing*



```
Stochastic Hill climbing distance :  1013.13 | Time  :  1.14
```

Fig 4(a): Route for 100 cities using Stochastic Hill Climbing for 10,000 iterations



```
Stochastic Hill climbing distance :  903.94 | Time  :  1.18
```

Fig 4(b): Route for 100 cities using Stochastic Hill Climbing for 20,000 iterations

V. Conclusion

In conclusion, Simulated Annealing performed better than all the other algorithms. There is a significant improvement in all algorithms when the iterations are increased. And the Stochastic Hill climbing gave close results to Simulated Annealing, Random Search Algorithm gave the worst results because of generating a random array at each time, the probability of generating an array that has the best path is highly not possible.

By increasing the number of cities then the number of iterations has to be increased which results in taking more computational time. By using Simulated Annealing with high iterations better results can be achieved compared to other algorithms.

References

[1] 1. https://en.wikipedia.org/wiki/Travelling_salesman_problem
[2] https://en.wikipedia.org/wiki/Simulated_annealing
[3] https://www.mathworks.com/help/gads/what-is-simulated-annealing.html
[4] Schrijver, Alexander (2003). Combinatorial Optimization: Polyhedra and Efficiency. Algorithms and Combinatorics. Vol. 24. Springer. ISBN 9783540443896.
[5] https://en.wikipedia.org/wiki/Annealing_(materials_science)
[6] https://api-ir.unilag.edu.ng/server/api/core/bitstreams/3a0c8880-b3d6-4f9a-83c8-5c1a2ac5621d/content
[7] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4808530/
[8] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.
[9] Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach (3rd ed.). Upper Saddle River, NJ: Prentice Hall.