IMPLEMENTATION AND STATISTICAL COMPARISON OF GENETIC ALGORITHM AND SIMULATED ANNEALING ALGORITHM TO SOLVE TRAVELING SALESMAN PROBLEM

Rushikesh Bagul
Msc Robotics
University of Birmingham
rxb278@student.bham.ac.uk

Abstract

This paper implements the genetic algorithm and simulated annealing algorithm to solve the Odyssey of Ulysses 22 cities Traveling Salesman Problem. This paper also discusses the steps to implement both the algorithms and steps to choose and tune the hyper parameters of both the algorithms. In the conclusion, the paper statistically compares both the algorithms using Wilcoxon signed-rank test to prove if one of this heuristic algorithms produces statistically significant results than the other.

1. INTRODUCTION

The Travelling Salesman Problem (TSP) is a famous problem in mathematics and computer science, which asks for the shortest possible route that visits a set of cities and returns to the starting city. The problem is a well-known example of an NP-hard problem. This problem is hard to solve because the solution space of the problem increases factorily with the increasing number of cities. If TSP is to be solved by the brute force algorithm its complexity is O(n!). Increasing the number of cities in the problem makes it increasingly hard to solve. This paper uses the Odyssey of Ulysses 22 cities Traveling Salesman Problem as a reference problem. This problem includes coordinates of 22 cities which need to be visited at least once in the route. The figure 1 shows the coordinates of the cities in the map.

Solving a problem like Odyssey of Ulysses 22 cities Traveling Salesman Problem[1] requires extensive computational power as well as time to find the optimum solution for the problem using brute force or dynamic programming. It may not be realistic to solve the TSP with a large number of cities. This will not only be very computationally heavy but also take a very long time to find the optimum solution. Other approaches to solve the large TSP in limited time are heuristic algorithms. Heuristic algorithms do not provide any guarantees about the quality of their solutions. Instead, they try to find good solutions in a reasonable amount of time. The solution obtained from the heuristic algorithm is not always optimal. But good enough to get the close approximation of the optimal solution in limited time. Heuristic algorithms are random in nature and produce different results every single time.

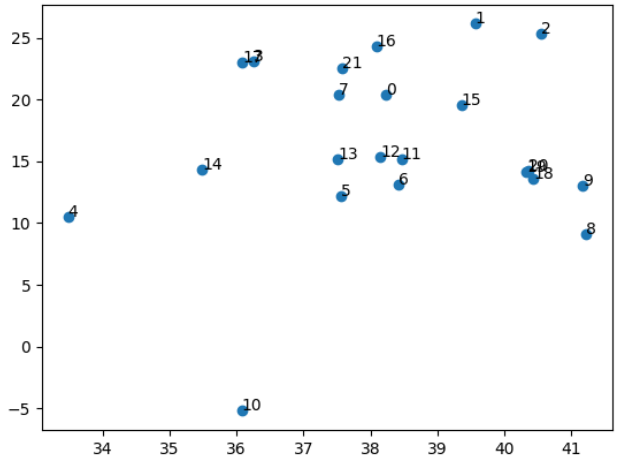

Figure 1 :- Coordinates of 22 cities in reference TSP

Two of the most famous heuristic algorithms are simulated annealing (SA) and genetic algorithm (GA). This paper implements both the algorithm to solve the reference tsp problem and then compares the results obtained from both the algorithms.

This paper is divided into 4 sections. Firstly it discusses the steps to implement both heuristic algorithms (SA, GA). In the second section it discusses the hyperparameters for both the algorithms and steps to tune these hyperparameters. Third section compares the results of both the algorithms using the Wilcoxon signed-rank test. Results of Wilcoxon signed-rank test are concluded in the final part of the paper.

All the implementation code used in this paper is available at https://github.com/rishifbagul/TSP_by_SA_and_GA

2. IMPLEMENTATION

2.1 SIMULATED ANNEALING

Simulated annealing algorithm inspired by the annealing process in metallurgy[2]. The algorithm is used to find the global optimum solution to a complex optimization problem by exploring the search space and gradually reducing the search space over time.

The simulated annealing algorithm begins with an initial random solution and then makes a series of small modifications to this solution. These modifications are made using a probability distribution that favors solutions that improve the objective function in this case reducing the distance of a route. However, unlike other optimization algorithms that always accept an improved solution, the simulated annealing algorithm sometimes accepts solutions that are worse than the current solution depending upon the hyper parameter temperature. This is done to avoid getting stuck in a local minimum and to explore the search space more broadly.

As the simulated annealing algorithm proceeds, the temperature parameter is reduced by the colling value and hence the probability of accepting worse solutions decreases over time, mimicking the cooling process in the metallurgical annealing process. This gradual cooling allows the algorithm to explore the search space thoroughly while still converging to a near-optimal solution. Algorithm 1 shows the pseudocode for the SA algorithm and figure 2 shows the flowchart.

Algorithm 1 Simulated Annealing SA(x0,T,C) initial_solution,temperature,cooling

```
1: x := x0; e := f (x )
2: xbest := x ; ebest := x
3: for i in range(max_iteration)
4:         T := T x C
5:         xnew := neighbor_solution (x)
6:         enew := objective_function(xnew )
7:         if P enew < e  then
8:                 x := xnew ; e := enew
9:         else if P (enew - e) / T  >R (0,1) then
10:                x := xnew ; e := enew
11:        if enew <ebest then
12:                xbest := xnew ; ebest := enew
13:Output xbest
```

Most important steps in the implementation of simulated annealing algorithm for TSP includes
1] represent tsp problem in a data structure
2]finding neighbor solution for the existing solution
3] objective function for evaluating the solution
4] temperature reduction function

In the implementation of TSP using simulated annealing this paper has used an array data structure to represent the ordered list of nodes to visit. It is very convenient to represent the TSP solution in an ordered list which holds the information about the order in which cities should be visited. For the initial solution a random permutation of a number 0 to 21 is initialized to be passed to the SA algorithm.

Neighbor function in this implementation takes the current solution which is a list of nodes and swaps any two nodes in that list randomly. Swapping two nodes in the list gives the new solution which is very similar to the current solution. Swapping of two nodes in the solution only changes two paths in the solution. Total euclidean distance between all the adjacent nodes in the solution is calculated in the objective function which represents the cost of the solution. In the TSP objective of the problem

is to reduce the objective function. It is a minimisation function.

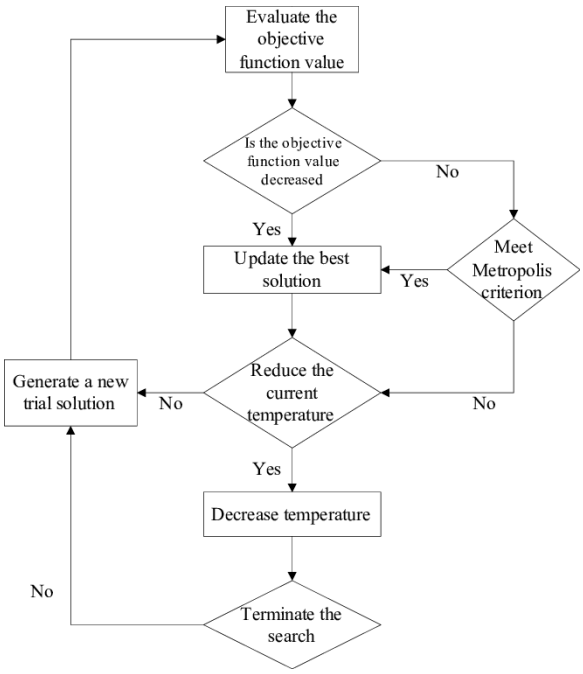

Figure 2 :- flowchart for simulated annealing

There are multiple functions available for reducing the temperature in the algorithm. The simplest one is linearly reducing the temperature. In this implementation linear reduction of temperature is used by multiplying temperature with the cooling rate.

In the final step of the algorithm when the number of iteration matches the maximum iteration number the best stored solution is printed with its cost value.

2.2 GENETIC ALGORITHM

A Genetic Algorithm (GA) is inspired by the principles of natural selection and genetics. The algorithm is used to find the global optimum solution to a complex optimization problem by mimicking the process of natural selection and the genetic crossover and mutation.

The GA algorithm begins with a population of candidate solutions, called chromosomes, that are randomly generated. Each chromosome represents a potential solution to the problem, and its fitness is determined by how well it solves the problem. The algorithm then selects the fittest chromosomes from the population and combines them through crossover and mutation operators to generate a new population of chromosomes.

Crossover involves combining two or more parent chromosomes to create a new offspring chromosome. This is done by selecting a random point in the chromosome and swapping the genes between the parents to create a new offspring chromosome. Mutation involves randomly changing one or more genes in a chromosome to create a new variation.

After the new population is generated, the fitness of each chromosome is evaluated, and with the reproduction function lease fit individuals are replaced in the next generation. This is an iterative process and the algorithm is run for a set amount of iterations. Algorithm 2 shows the pseudocode for the GA.

Algorithm 2 Genetic Algorithm

1: X_0 := generate initial population of solutions
2: V_0:=Evaluate fitness of each individual in X_0.
3: **for** i in range(max_iteration)
4: Selection:Select parents from Xt based on fitness.
5: Crossover:Crossover the selected parents to breed
6: Fitness: Evaluate the fitness of new individuals.
7: Reproduction: generate new population by replacing least fit parents
8:**Output** Xbest

The implementation of genetic algorithms involves several crucial steps, including encoding the problem in genotype, selecting the appropriate operator, determining the crossover and fitness functions, and defining the reproduction function. To apply genetic algorithms to the Traveling Salesman Problem (TSP) in this study, a simple encoding scheme was adopted whereby the solution of the route is represented as an ordered integer list of nodes ranging from 0 to the maximum number of nodes. The ordered list efficiently conveys the order in which the cities should be visited. The initial population was generated by randomly permuting a sequence of numbers from 0 to 21.

The choice of selection operator significantly impacts the performance of the algorithm by balancing the trade-off between exploration and exploitation. Therefore, this study employed the commonly used binary tournament operator to randomly select two parents based on their fitness and then crossbreed them to produce new offspring.

However, typical crossover functions such as binary and multicut crossover are not suitable for TSP due to the chosen encoding scheme, which prohibits the presence of the same node twice in the solution. Instead, a Partially Mapped Crossover Operator[3] was utilized, which randomly cuts the parents to transfer the sequence of nodes to the offspring and then fills the remaining spots in the sequence based on the parent's order. This crossover function creates two children by crossing the features of the two parents.

To optimize the solution for the TSP, the fitness function selected in this study is the inverse of the total distance required to complete the route. The goal is to minimize the distance traveled, thereby increasing the fitness of the individual. The reproduction function plays a critical role in the genetic algorithm as it generates a new population based on the fitness values of the parents and offspring. This paper employs a strategy in which the percentage of selection based on the fitness value is determined, and the remaining individuals are selected at random. The subsequent section discusses the effects of altering the selection percentage.

3. HYPERPARAMETER SELECTION

3.1 HYPERPARAMETERS OF SIMULATED ANNEALING

To achieve optimal results in the simulated annealing algorithm, two critical parameters require selection and fine-tuning - the initial temperature and the cooling rate. The fundamental concept behind selecting the initial temperature and cooling rate is to balance the exploration and exploitation stages of the algorithm. At the outset, the algorithm must enable greater exploration and choose even the suboptimal solutions. However, in the later stages, it should decrease the probability of selecting the inferior solutions and focus on exploiting the search space to reach the global minimum. Therefore, it is essential to determine a combination of initial temperature and cooling rate that produces these desired outcomes.

Selection of initial temperature is based on the paper [4]. It gives the method to select the initial temperature. Choose tau0 = 0.5 as assumed quality of initial solution (assuming a bad one), and deduce T0 from exp(-fx_mean/T0) = tau0, that is, T0 = -fx_mean/ln(tau0). From the method given in the article and calculating -fx_mean as 100 initial temperatures calculated turned out to be 144.
Initial_temeperature= 144

Selecting an appropriate cooling rate is vital to ensure that the temperature and selection probability decrease gradually, thereby balancing the exploration and exploitation of the search space. To identify the optimal cooling rate, 20 individual runs were conducted for each rate, and the readings were recorded. Refer table 1.

Sr.no	Cooling rate	Average distance over 20 runs
1	0.9970	81.54
2	0.9980	78.12
3	0.9990	77.75
4	0.9995	80.43

Table 1: Effect of cooling rate on results

It is clear from the experiment runs that cooling rate 0.9990 is producing the best results on average and hence it is chosen to be the final cooling rate.
Cooling_rate=0.9990
Figure 3 shows the effect of cooling rate on the selection probability over the run of 10000 iterations. The graph on the y axis shows the selection probability for constant energy vs the iteration number on the x axis. It is clear from the observation that the cooling rate 0.9970 and 0.9980 prematurely saturate to 0 and does not

allow the exploration of the solution. While cooling rate 0.9995 fails to accept the high energy solution early on and doesn't have enough time in the second part to exploit the solution. Cooling rate 0.9990 balances both the exploration and exploitation part well. Allowing the inferior solutions early on with probability close to 0.9 and then leaves over enough time to exploit the best solution in the later part.

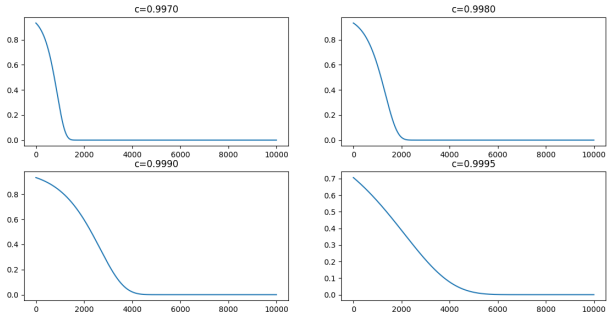

Figure 3:- Effect of cooling rate on selection probability

3.2 HYPERPARAMETERS OF GENETIC ALGORITHM

In genetic algorithms two of the most important choices available are whether to mutate the children or just use them based on the crossover of their parents. In this paper we have used the mutation of swapping two nodes after the crossover function of the parents. Following table 2 shows the effect of mutation over the result of solution.

Sr. no	Mutation	Average distance over 10 runs
1	No	91.88
2	Yes	78.52

Table 2:- Effect of mutation on result

Mutation: Yes
It is very clear from the observation that the mutation helps diversify the solution space in the generation. Not being able to mutate and just using crossover functions makes the children very similar to their parents and reduces the diversity in the population.

Another important factor in the selection of the genetic algorithm is population size. Table 3 shows the effect of population size on the result.

Sr. no	Population	Average distance over 10 runs
1	20	79.39
2	50	76.95
3	100	78.43

Table 3:- Effect of Population size on result

Population Size= 50
It is obvious from the observation that increasing the population reduces the total number of generations we can evaluate in a fixed number of objective function calculations. Increasing the number of population sizes reduces the generation number and hence the solution doesn't get enough time to evolve and optimize. However, reducing the number of population also adversely affects the result because reducing the population number does not allow the variety of solutions in the generation.

Another factor to choose in the genetic algorithm is reproduction percentage. This controls the number of individuals selected in the next generation based on the objective function and number of selection based on the random basis. Percentage of reproduction can be seen as a way to explore and exploit the solution. Table 4 shows the effect of reproduction percentage on the result.

Sr. no	Percentage	Average distance over 10 runs
1	10%	78.46
2	50%	77.92
3	90%	77.09

Table4:- Effect of reproduction percentage on result

It is obvious from the observation that increasing the reproduction percentage allows the best

performing solutions in the generation and exploiting the search space. Reducing the number allows the not optimized solution to be in the search space.

Reproduction percentage:- 90%

4. RESULTS

For the experimentation and comparison both SA and GA algorithms were run for 30 trials with a maximum number of objective function calculations of 10000. That is 10000 iterations for the SA and 200 generations of 50 population for GA. Results recorded for the 30 trials are highlighted in the table5.

	SA	GA
Best Distance	75.6666	75.53561
Worst Distance	80.69715	88.36397
Mean Distance	77.6955	77.7617
Standard Deviation	1.423295	2.413625

Table 5:- Results for SA and GA in 30 runs

Figure 4 shows the solutions for all the 30 runs of SA with their distance value. Figure 5 shows the best distance found by SA in all 30 runs. Figure 6 Shows the solution for all the 30 runs of GA with their distance value. Figure 7 shows the best distance found by GA in all 30 runs.

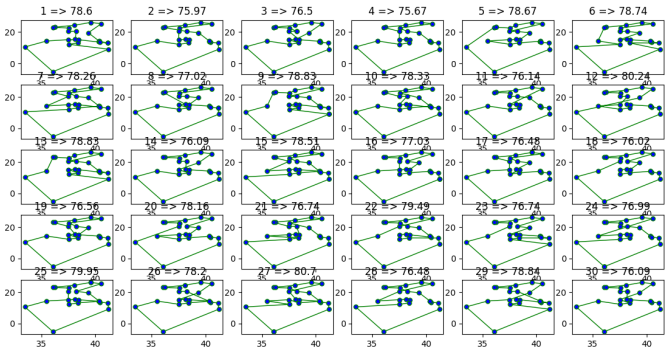

Figure 4:- Results of all 30 runs of SA

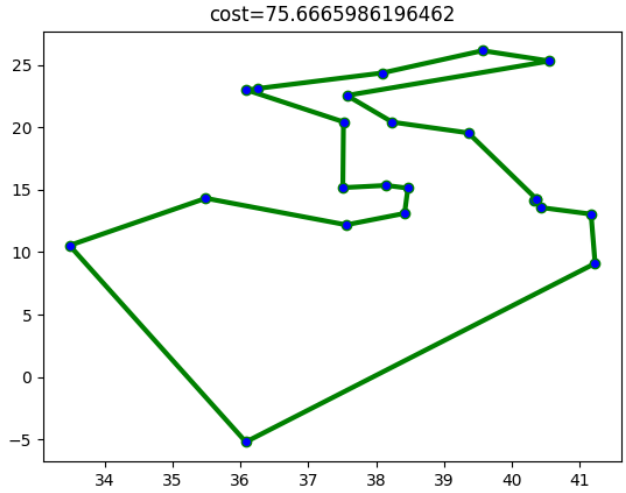

Figure 5:- Best result for SA in 30 runs

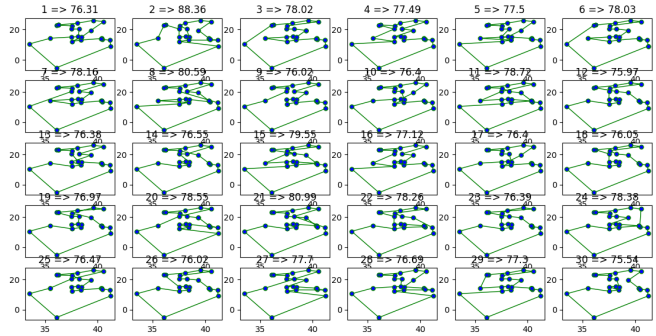

Figure 6: Results of all 30 runs of GA

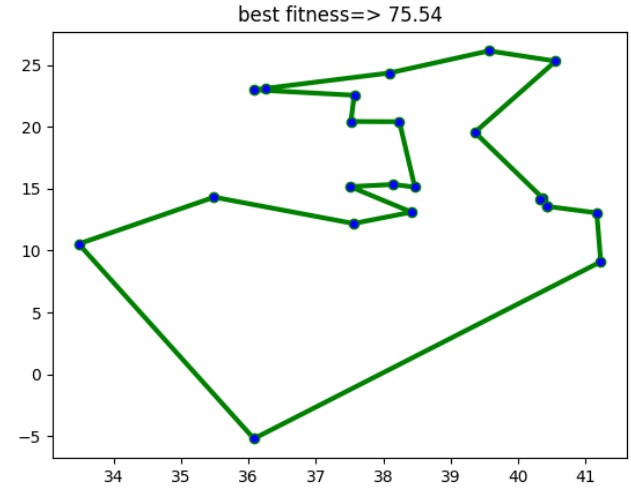

Figure 7:- Best result for GA in 30 runs

5. STATISTICAL TEST

To compare the results of SA and GA statistically let μ_1 = Mean of the SA results and μ_2 = Mean of the GA results. Then the Null Hypothesis is that no algorithm is statistically better than the other and both μ_1 and μ_2 are drawn from the same normal distribution. Mathematically ,

Null Hypothesis : H0: $\mu_1 = \mu_2$
Alternate Hypothesis: H1: $\mu_1 \neq \mu_2$

Table 6 calculates the Wilcoxon Signed-rank test on the results of SA and GA algorithms.

No	SA	GA	Difference	Sign rank
1	78.59839	76.31187	2.28651359	21
2	75.97084	88.36397	-12.393125	-30
3	76.49977	78.0247	-1.5249331	-16
4	75.6666	77.4898	-1.8232005	-18
5	78.67217	77.50023	1.17193661	13
6	78.74143	78.0253	0.7161233	11
7	78.26241	78.16297	0.09943414	4
8	77.02435	80.58828	-3.5639263	-27
9	78.82589	76.01877	2.80711609	24
10	78.33299	76.39799	1.93499938	19
11	76.13596	78.7182	-2.5822417	-23
12	80.23567	75.97084	4.26482537	29
13	78.82589	76.37566	2.45023049	22
14	76.088	76.54558	-0.4575802	-9
15	78.51365	79.54826	-1.0346018	-12
16	77.0315	77.12041	-0.0889038	-3
17	76.4839	76.39799	0.08590814	2
18	76.01817	76.04853	-0.0303671	-1
19	76.56457	76.96687	-0.4023038	-8
20	78.16297	78.5515	-0.388527	-7
21	76.74141	80.98536	-4.2439572	-28
22	79.4934	78.25589	1.2375103	14
23	76.73515	76.39491	0.3402402	6
24	76.99102	78.38217	-1.3911515	-15
25	79.94718	76.46797	3.47921118	26
26	78.19846	76.02042	2.17803593	20
27	80.69715	77.69824	2.99891255	25
28	76.47773	76.68629	-0.2085593	-5
29	78.84034	77.29651	1.54382881	17
30	76.088	75.53561	0.55238465	10

Table 6:- wilcoxon signed rank for all 30 results

T+ =263
T- =202
W=min(T+,T-)= 202
For
Level of confidence =c=0.95
Level of significance= α= 1-c=0.05

Critical value $W_{Critical}$ = 137 …Lookup table[5]
W=202 > $W_{Critical}$ = 137

We fail to reject the Null Hypothesis.

To use a two sided Wilcoxon signed rank test we firstly calculated the difference between the results of SA and GA. then we rank the absolute value of the difference. Rank is then multiplied with the sign of the difference. W is calculated from the T+ and T- values. Above calculation uses the level of significance of 0.05 for testing the hypothesis.
Using the above calculation for the Wilcoxon signed rank test it is clear that There is not statistically significant difference between the two algorithms. Mean value of both the

algorithms are also very close. However the standard deviation of GA is higher than the SA.

6. CONCLUSION

This paper tests the Simulated Annealing and Genetic Algorithm to Solve the Odyssey of Ulysses 22 cities Traveling Salesman Problem. It discusses the multiple hyperparameters for both the algorithms. It uses a Wilcoxon signed-rank test to compare both the algorithms for solving the TSP problem. After the Statistical comparison it concludes that there is no statistically significant difference in Simulated Annealing and Genetic Algorithm for solving traveling salesman problem.

7. REFERENCES

[1]The Optimized Odyssey by Martin Grˆtschel and Manfred Padberg

[2]Optimization by Simulated Annealing by S. Kirkpatrick, C. D. Gelatt, M. Vecchi

[3]Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator by Abid Hussain,Yousaf Shad Muhammad,1M. Nauman Sajid,Ijaz Hussain,Alaa Mohamd Shoukry,and Showkat Gani

[4]Dréo, Johann, et al. Metaheuristics for hard optimization: methods and case studies. Springer Science & Business Media, 2006.

[5]Wilcoxon Signed Rank Test https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_nonparametric/BS704_Nonparametric6.html