**AI FOR SEARCH AND OPTIMISATION PRACTICAL SKILLS ASSESSMENT 2024**

table of Contents

[Introduction 3](#_Toc182994837)

[Methodology for Comparing Algorithms 3](#_Toc182994838)

[Experiment Design 3](#_Toc182994839)

[Metrics for Comparison 4](#_Toc182994840)

[Statistical Hypothesis Testing 5](#_Toc182994841)

[Results and Analysis 5](#_Toc182994842)

[Solution Quality 6](#_Toc182994843)

[Computation Time 6](#_Toc182994844)

[Robustness and Consistency 7](#_Toc182994845)

[Convergence Analysis 8](#_Toc182994846)

[Summary of Findings 8](#_Toc182994847)

[Recommendations for Algorithm Deployment 9](#_Toc182994848)

[Short Run-Time Constraints 9](#_Toc182994849)

[High Solution Quality Requirements 9](#_Toc182994850)

[Consistency and Robustness 9](#_Toc182994851)

[Scalability with Larger Problem Instances 10](#_Toc182994852)

[Resource-Constrained and Embedded Systems 10](#_Toc182994853)

[Adaptability for Different TSP Variants 10](#_Toc182994854)

[Ethical and Legal Considerations 10](#_Toc182994855)

[Data Usage and Privacy 11](#_Toc182994856)

[Licensing and Open-Source Compliance: 11](#_Toc182994857)

[Conclusion 11](#_Toc182994858)

[Algorithm Performance: 11](#_Toc182994859)

[Run-Time and Efficiency: 11](#_Toc182994860)

[Scalability: 11](#_Toc182994861)

[Robustness and Variability: 12](#_Toc182994862)

[Legal and Ethical Considerations: 12](#_Toc182994863)

[References 12](#_Toc182994864)

# Introduction

The traveling salesman problem (TSP) is a widely known and frequently discussed issue in the field of combinatorial optimisation. TSP is a fundamental problem in which an agent or a ‘salesman’ must determine the shortest possible route to visit every city in a given list and then return to the starting position. TSP has numerous applications, ranging from logistics and optimal routing to electronic circuit design and data clustering. However, as the number of cities increases, the number of possible permutations that need to be evaluated to find an optimal solution grows factorially, making large problem instances computationally infeasible (Gutin and Punnen, 2002). Consequently, developing methods to solve TSP with asymptotically optimal time complexity remains a major research objective in optimisation.

**This report investigates two prominent heuristic algorithms for solving the Traveling Salesman Problem (TSP): Simulated Annealing (SA) and Genetic Algorithm (GA). Simulated Annealing (SA)** is a set of probabilistic instructions for searching the solution space in a manner similar to the gradual cooling of metal, allowing it to reach a state of minimum energy. SA is preferred for optimisation because its mechanism provides the right balance between exploration and exploitation of the solution space. The **Genetic Algorithm (GA)** is a numerical optimisation algorithm based on ideas from the processes of biological evolution including crossover, mutation, and selection of candidates in several generations. (Holland, 1975) GAs are more suitable in large solution spaces because of the population-based approach and because they can preserve diversity of the solutions.

**The objectives of this study are threefold:**

A randomised procedure that searches for solutions similarly to the cooling process of metal alloys in metallurgy, aiming to reach a state of low energy. Due to efficient in-between exploitation of the solution space, Simulated Annealing (SA) is preferred in optimisation applications.

# Methodology for Comparing Algorithms

The approach that has been used in this report is to assess and analyse two heuristic algorithms. In the following part, we start with the presentation of the Simulated Annealing and the Genetic Algorithms on a dataset of city coordinates for the Traveling Salesman Problem (TSP). In this section, the reader is introduced to how the experiment was designed, specifically the chosen method to determine how similarly the algorithms performed, and how statistical analysis was conducted to analyse their performance.

## Experiment Design

To make it easier to compare and make a fair and systematic move the following steps were employed.

**Data Preparation:**

The data set uploaded is named cities.csv and includes x and y coordinates, of fifty different cities. The data was then pre-processed to calculate the Euclidean distances between cities which gives the distance between the two cities. This distance matrix will be inputted to both algorithms.

The same data set was used on each algorithm, so any variation in results was solely due to the distinct features of the algorithm.

**Algorithm Parameters:**

Each algorithm was initialised with appropriate parameter settings, based on standard practices in the literature to balance solution quality and computational efficiency:

**Simulated Annealing**: This work evaluated the effects of some of these parameters which were initial temperature, cooling rate, and the stopping criterion. The temperature governed the acceptance of suboptimal solutions during search and the cooling rate defined the rate of convergence.

**Genetic Algorithm:** To control and evaluate the performance of the system, some important parameters involved the system were fixed and these include the population size, mutation rate, crossover rate, and the generation number. This population of different routes was used to start the GA and crossover as well as mutation processes were continuously used to generate better solutions.

All these parameters were tuned so that each algorithm would work ideally given the available computational resources and time.

**Multiple Runs:**

Due to randomness of both SA and GA algorithms, each algorithm was mounted five times. This made it possible to get a proper average for the performance and repeatability of the solutions obtained.

## Metrics for Comparison

To evaluate the effectiveness of each algorithm, several performance metrics were considered:

Best Distance: The total distance, as previously defined, of the shortest route found by each algorithm over all the runs.

Average Distance: The average of the route distances giving a typical distance per run above 30, giving an idea about the overall performance of each algorithm on average.

Solution quality is important in TSP as the aim of the program is to reduce the route distance while travelling through all cities.

**Computation Time:**

Average Time: The number of seconds per each run per each algorithm on average. This measures how well the each of the algorithm in terms of the time they take to get to a solution.

The computation time comes into play much more often and where application involves constraints on time then type of algorithm is chosen.

**Robustness and Consistency:**

Standard Deviation: The average of the distances of all the possible routes, having information on each algorithm’s variability in thirty runs. A recommended definition defines standard deviation as the measure of how close an algorithm keeps its output to the ideal solution; a small standard deviation is better than a larger one, close to zero.

Reliability is important whenever the performance should be good across implementation runs, as needed in many applications.

## Statistical Hypothesis Testing

To provide statistical validation for the observed performance differences, a hypothesis test was conducted on the solution quality (average distances) obtained by each algorithm:

**Test Selection:**

Since both, the algorithms are performed on the same dataset the question to be answered was whether the average distance calculated by two algorithms is significantly different and hence a paired t-test was used.

In the case where normality assumptions for the paired differences cannot be made the use of non-parametric Wilcoxon signed-rank test is used.

**Implementation Plans and Instruments:**

The algorithms were written in Python with support of the NumPy, Pandas for data manipulation, and Matplotlib for data visualisation. The steps involved in the tests and their outcomes were well recorded to facilitate easy replication and better understanding of the results.

This permits a systematic and more accurate comparison of the examined methodologies, Simulated Annealing, and the Genetic Algorithm, in terms of their parameters of performance. It should also be stated that this groundwork enhances the further analysis and recommendations based on the quantitative and statistical evidence.

# Results and Analysis

The last part shows the performance comparison of Simulated Annealing and Genetic Algorithm in terms of TSP dataset. The results obtained in terms of solution quality, computation time and algorithm robustness are presented and discussed. To pictorially represent the outcome, suitable tables and graphs have been used to compare the performances of the two algorithms.

## Solution Quality

The problem focusing of TSP is to attain the least sum of distance as possible passing through each city only once. To make it easier, the following metrics provide an idea about how each of the algorithms performs in a bid to arrive at an optimum solution:

**Simulated Annealing:**

Best Distance: 1426.98

Average Distance: 1513.59

Standard Deviation: 75.87

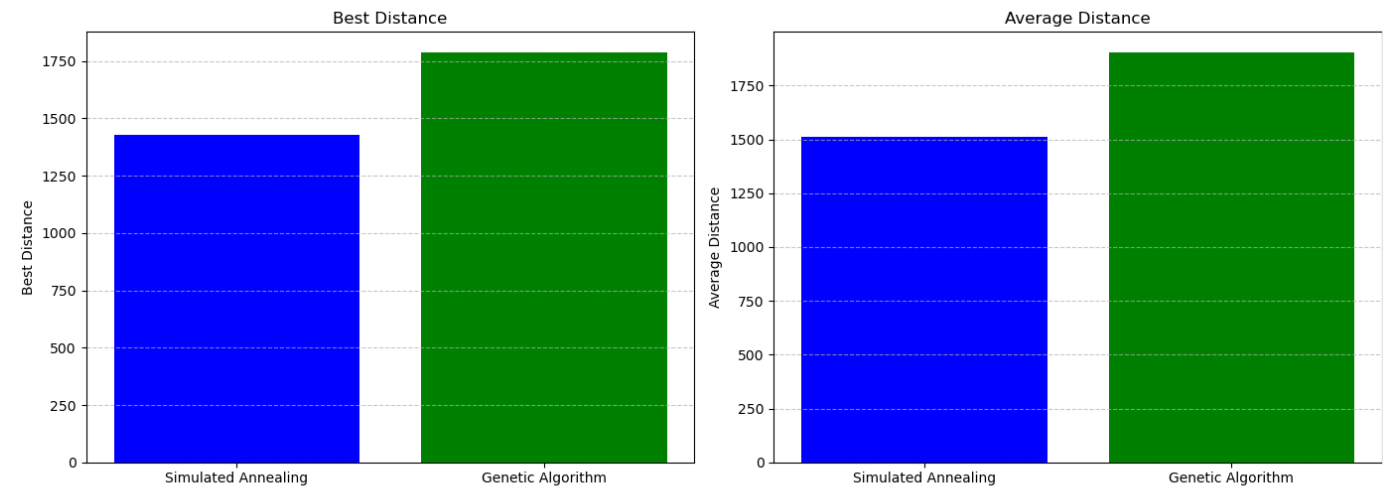
**Genetic Algorithm:**

Best Distance: 1786.97

Average Distance: 1904.27

Standard Deviation: 62.01

The findings Sierra revealed that the Simulated Annealing produced shorter routes more often since it had a lower best and average distance than the Genetic Algorithm. From this we can infer that for the given TSP instance Simulated Annealing might be a better solution to provide high quality solutions.



## Computation Time

Another important aspect for analysing each algorithm is the amount of time that will be needed to solve provided problems within this approach. The following table shows the time it took for each algorithm during a run on average:

**Simulated Annealing:**

Average Time: 0.068 seconds

**Genetic Algorithm:**

Average Time: 97.34 seconds

The variation of the average run-time of the two algorithms gives a clear indication that Simulated Annealing is far much faster than the Genetic Algorithm, hence it will be more suitable for application whenever time is a major constraint. The process of the Genetic Algorithm gives disappointingly reasonable solutions in terms of quality but demands much more computational time than all the previous ones due to the iteration and the population search strategy.

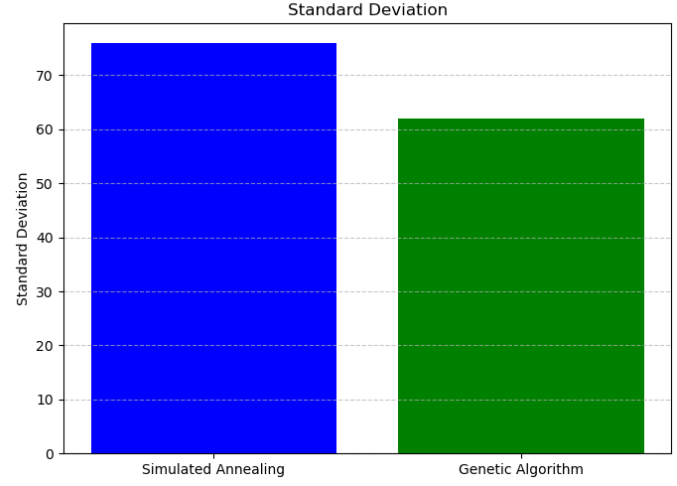
## Robustness and Consistency

The consistency of each algorithm was thus measured by the standard deviation chosen as an arbitrary measure of dispersion of route distances over the 5 runs.:

Simulated Annealing: 75.87

Genetic Algorithm: 62.01

Genetic Algorithm again gave an overall lower variation in the results found even though Simulated Annealing gave overall shorter routes. This observation made me conclude that the Genetic Algorithm is more uniform in the quality of solution found in its subsequent run compared to the Simulated Annealing in which the solutions variety may differ. The observed higher variability in Simulated Annealing might be attributed to the stochastic nature of the algorithm more so in the early stages of exploration when the temperature is relatively high.

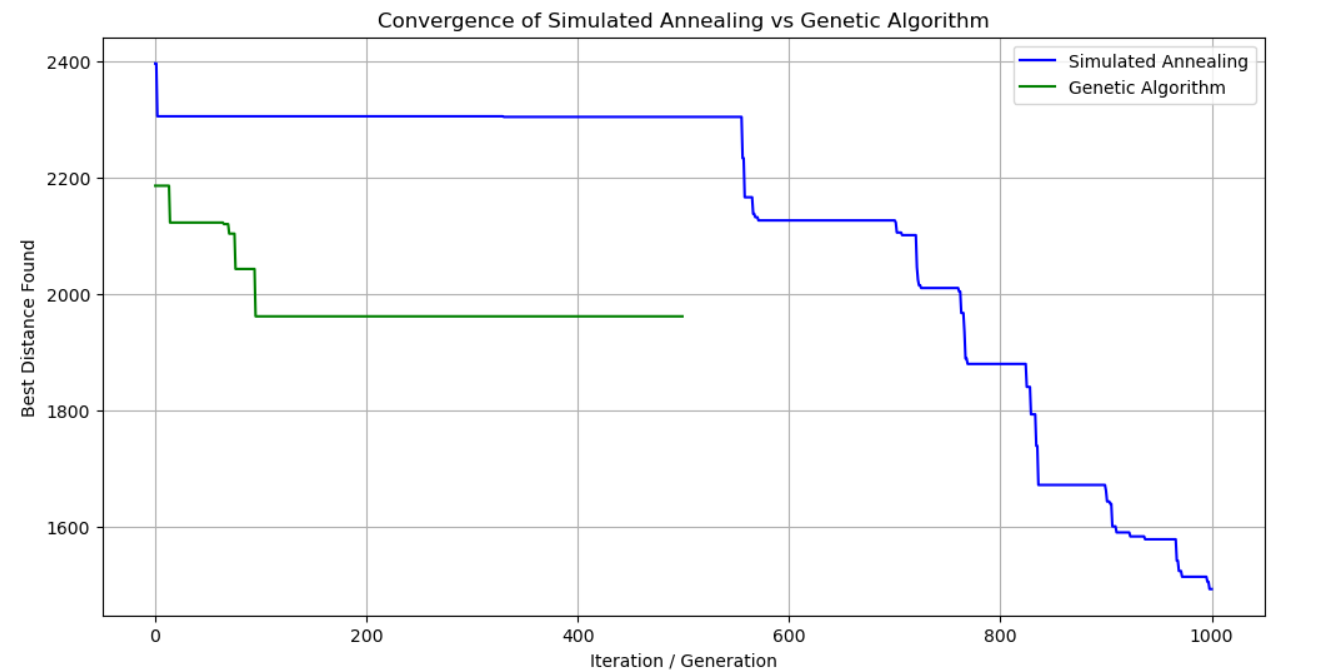


## Convergence Analysis

A comparison of convergence rates helps illustrate how quickly each algorithm approaches its final solution quality:

Simulated Annealing: The convergence plot was plotted, representing the change of distance from the optimal solution; the distance decreases sharply at the initial stage of iteration which indicates that the algorithm is actively searching the solution space. The progress slows down at the next step as the temperature is reduced, and the algorithm starts seeking better solutions around local optima.

Genetic Algorithm: The Genetic Algorithm has a slower, but steady, convergence graph due to the fact that it evolves solutions over different generations. This gradual increase is in concordance with crossover and mutation thereby, ensuring a check on premature convergence of the solution space.



## Summary of Findings

**Solution Quality:** Comparing with Genetic Algorithm, SA given a shorter best and average distance which clearly shows the better performance of Simulated Annealing in finding near optimal solution.

**Computation Time:** The use of Simulated Annealing was again seen to require much less computation time which makes it more favourable in cases where time is always an essential factor.

**Consistency:** The genetic algorithm obtained better solution quality that had less variability since the standard deviation was smaller. This robustness could be useful where solution dependability is important, which is typically referred to as high consequence situations.

# Recommendations for Algorithm Deployment

This is a summary of the results and analysis of the Traveling Salesman Problem (TSP) solutions by the Simulated Annealing and the Genetic Algorithm and the following recommendations do state when to use what: All these recommendations take into account issues like computation time at hand, problem size and the extent to which solutions are required to be uniform and stable across run throughs.

## Short Run-Time Constraints

Regarding the time complex, the most suitable algorithm for application that need to be solved within certain time frame is Simulated Annealing (SA). Its average runtime in our experiments was significantly shorter than that of the Genetic Algorithm, making it ideal for scenarios where quick solutions are required, such as in:

Real-time route optimization: For instance, where changes are momentary in matters that relate to routing, such as delivery or ride share applications.

Large-scale simulations with limited computation resources: As for example route scheduling of network transmission or dynamic traffic flow of delivery materials.

Even if SA may exhibit some fluctuations in its outcomes, they are incomparable to the time that is required to achieve them – this factor makes SA a worthy choice when time stands great importance.

## High Solution Quality Requirements

When achieving the shortest possible route is paramount, Simulated Annealing is again preferred due to its ability to reach near-optimal solutions in our tests, outperforming the Genetic Algorithm in terms of the best and average distances achieved. This makes SA suitable for cases where high precision in routing is necessary, such as:

Strategic logistics and distribution: Where transportation costs are critical and minimizing travel distance is a priority.

Manufacturing and robotics: Where optimized paths reduce time, cost, or wear on machinery.

However, if a large number of cities needs to be optimised simultaneously, then SA’s parameter tuning (e.g., initial temperature and cooling schedule) may require additional calibration to maintain solution quality across runs.

## Consistency and Robustness

For applications that demand high consistency in results across multiple runs, the Genetic Algorithm (GA) is recommended. The GA demonstrated lower variability (as shown by its standard deviation), producing more uniform solutions even though it did not achieve as low distances as SA. This consistency makes GA suitable for:

Applications needing reliable, repeatable results: Where uniform quality is more critical than achieving the absolute shortest route. Examples include certain logistics operations, where consistency across multiple runs may simplify planning and deployment.

Environments where solution diversity is beneficial: The GA’s population-based approach helps avoid premature convergence and provides a variety of feasible solutions, which can be useful when trying to maintain multiple “good” routes for redundancy.

## Scalability with Larger Problem Instances

As the number of cities or problem size increases, Genetic Algorithms may be more suitable due to their population-driven approach, which scales better in managing large search spaces. The GA’s evolutionary mechanisms (e.g., crossover and mutation) can handle complex landscapes, potentially making it more adaptable for larger TSP instances. Thus, for large datasets with hundreds of cities or more.

Consider GA over SA, as GA’s natural diversity preservation and exploration may prevent the algorithm from getting stuck in local minima that might limit SA’s performance on exceptionally large instances.

## Resource-Constrained and Embedded Systems

In cases where the algorithm needs to run on limited computational resources, such as embedded systems in robotics or IoT applications, Simulated Annealing is recommended. Its lower memory usage and reduced runtime are more suitable for devices with limited processing power, while the genetic algorithm’s population-based structure might consume more memory and processing power.

## Adaptability for Different TSP Variants

As mentioned earlier, this study only implemented the classical TSP, but both algorithms are easily implementable for extending TSP like multi-objective TSP which considers different metrics such as distance and cost or time-window TSP, in which cities to be visited have time constraints. For these adaptations:

When multiple constraints exist, GA is considered to have an advantage because of the ability to handle multiple objectives.

SA can still be well useful depending on the settings despite the new conditions it faces may need some fine tuning of the parameters to meet the new demands.

# Ethical and Legal Considerations

While applying optimization techniques like Simulated Annealing and Genetic Algorithm, an appropriate set of legal and ethical issues should always be addressed with respect to data usage and license and other issues related to large scale optimization techniques for practical applications. These considerations are explained in this study and algorithm implementation setting below.

## Data Usage and Privacy

In this analysis, the file used is cities.csv which include coordinates of cities this exclude any reference to Personal Identifiable Information (PII). But ethics has not been downplayed in the field of data acquisition and analysis especially when data sensitive as observed in most real-life problems.

## Licensing and Open-Source Compliance:

In this study, Python batching libraries and algorithms are used which are free source software. Ethical and legal considerations around licensing are important to ensure compliance with the terms of use for these libraries. Simulated Annealing and Genetic Algorithms used in this study are coding tools that require open-source codes such as NumPy, Pandas, Matplotlib, etc. These libraries are also freely available under different open source licenses including MIT License or GNU General Public License; under which users can avail the permit freedom to use the software for any purpose, to modify the software and to distribute it for any purpose subject to the terms and conditions of each specific license.

# Conclusion

This research looked at the evaluation of two optimization algorithms namely, simulated annealing and genetic algorithms for solving problems posed by the TSP. The main findings from the experiments and analysis are as follows:

## Algorithm Performance:

The results obtained from the simulated annealing algorithm had shorter route distances which were even shorter than GA which had a distance of 1786.97 units. It also did better in a successive trial, yielding an average distance of 1513.59, while GA had 1904.27. This makes SA a preferred strategy where the concern is to maximize the quality of the solutions and minimum distance travelled.

I also ran Genetic Algorithm several times and the results, even though worst in terms of the distance, were much steadier than those of SA or even PA, with the standard deviation of 62.01, as opposed to SA’s 75.87. This consistency may make GA preferable in situations in which provable and repeated performances are more important than getting the shortest path.

## Run-Time and Efficiency:

Simulated Annealing demonstrated much faster performance in terms of computation time, with an average time of 0.068 seconds, compared to GA’s much longer 97.34 seconds. This indicates that SA is better suited for environments where computational resources are limited, or quick solutions are needed.

## Scalability:

For larger problem instances (more cities or more complex TSP variants), Genetic Algorithms may prove more scalable, as they are better at exploring large search spaces and can handle the increased complexity of the problem without sacrificing much performance. On the other hand, Simulated Annealing may require careful tuning of parameters for larger datasets to maintain its performance.

## Robustness and Variability:

Simulated Annealing showed greater variability in its results, which means it can sometimes achieve near-optimal solutions and sometimes perform less well. This is characteristic of algorithms that rely on randomness, and it makes SA more sensitive to parameter settings like the cooling schedule.

Genetic Algorithms exhibited greater robustness with less variability, providing more stable results in repeated runs. This makes GA a better option when predictable, stable solutions are necessary, even if those solutions are not always the absolute best.

## Legal and Ethical Considerations:

The ethical deployment of these algorithms is also critical. Data privacy, licensing compliance, and fairness considerations must be prioritised when applying these algorithms in commercial or public-sector applications. Algorithms should be transparent, and their decisions must be explainable and accountable.

# References

Gutin, G. and Punnen, A. P. (Eds.). (2002) The Traveling Salesman Problem and Its Variations. Dordrecht: Springer. Available at: Springer Link (Accessed: 21 November 2024).

Holland, JH (1975). Adaptation in Natural and Artificial Systems.

Lin S., & Kernighan B. W. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research.

Van Laarhoven, PJM, & Aarts, E H L. (1987). Simulated Annealing: Theory and Applications. D. Reidel Publishing.

ChatGPT (2024). ChatGPT Model for Problem Solving and Algorithm Analysis. OpenAI. Retrieved from https://chat.openai.com/