AI FOR SEARCH AND OPTIMISATION
PRACTICAL SKILLS ASSESSMENT 2024

Contents

1. 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 optimization.

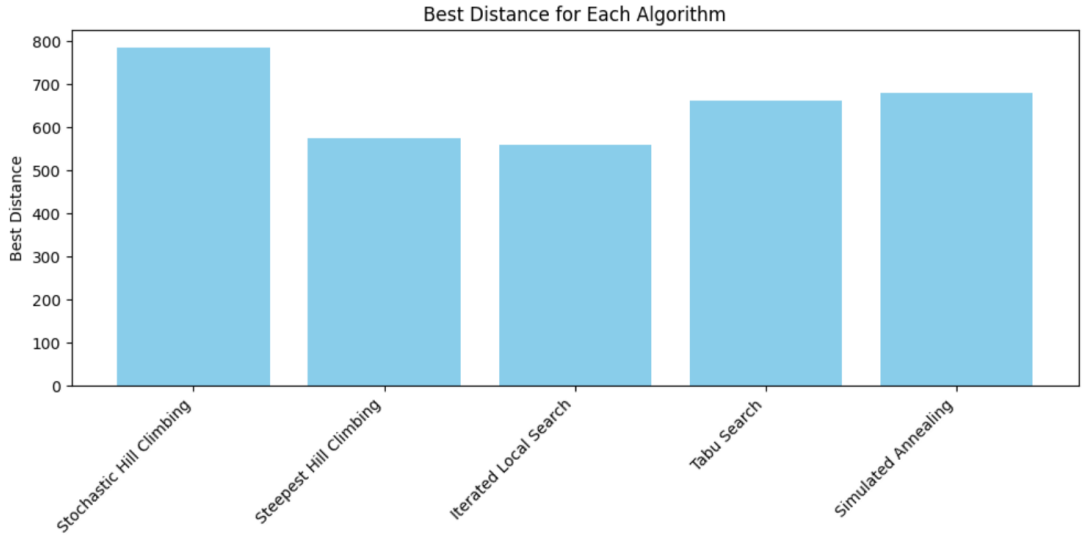
This report explores two heuristic approaches to solve TSP: **Simulated Annealing (SA)**, a single-driven local search algorithm, and **Genetic Algorithm (GA)**, a population-driven evolutionary search algorithm. The aim is to compare these techniques in terms of solution quality, computational efficiency, and robustness. By analyzing the results on a dataset of city coordinates, this report sheds light on the strengths and limitations of each method.

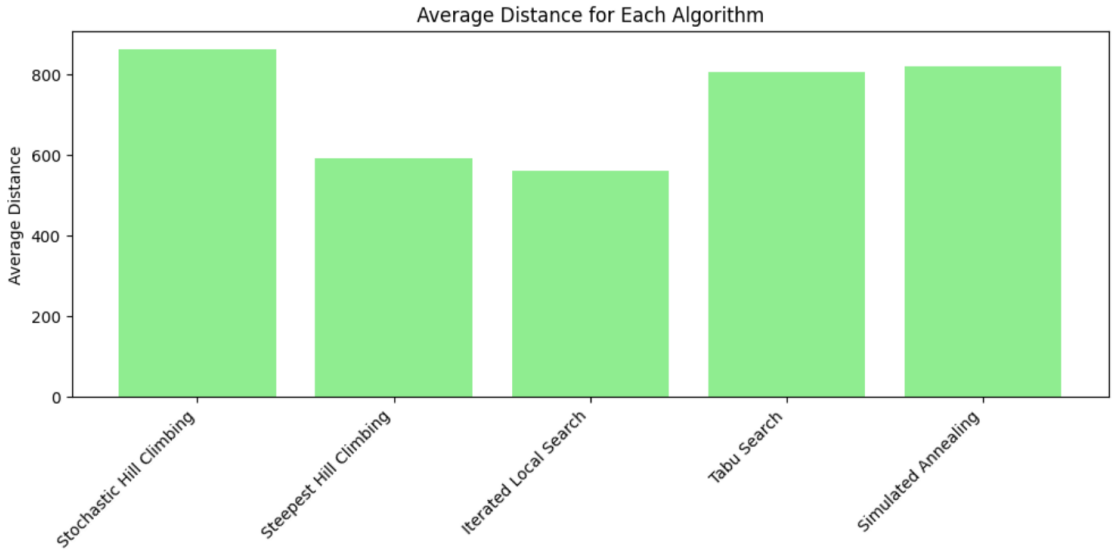
1.1 Single Solution Driven Search Algorithms

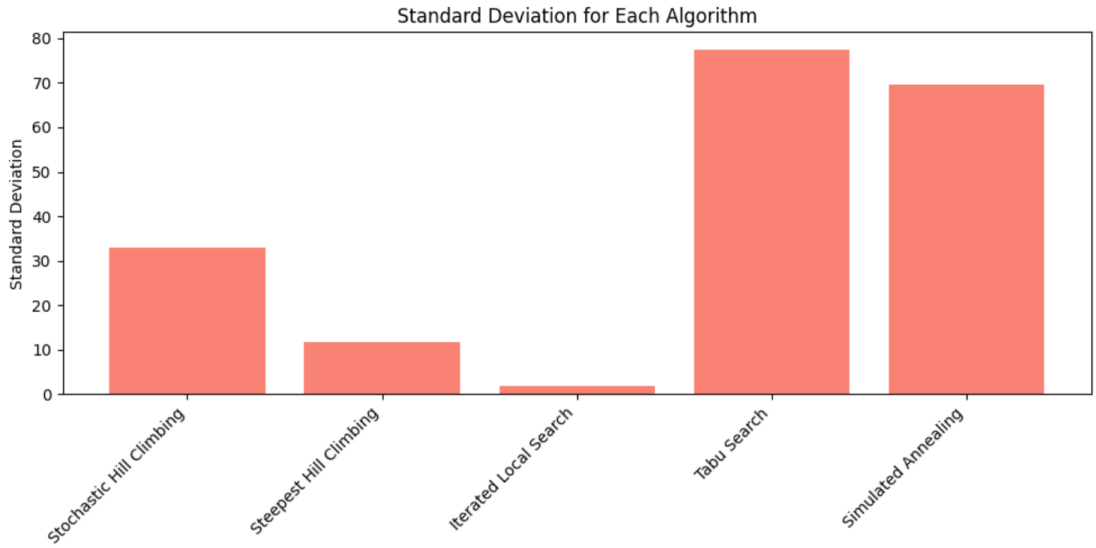
Single-driven search algorithms operate by focusing on a single solution at any given point in time. These algorithms iteratively improve upon the current solution by exploring neighboring states or applying specific transformation rules. Examples include:

- **Hill Climbing:** Starts with an arbitrary solution and iteratively moves to a better neighboring solution until no improvements can be found. However, it often gets stuck in local minima. (Rossi, et al., 2006)
- **Simulated Annealing (SA):** Combines the iterative improvement process of Hill Climbing with a probabilistic mechanism to escape local minima. This is achieved by accepting worse solutions based on a probability that decreases over time. (Czech & Czarnas, 2002)
- **Tabu Search:** Enhances Hill Climbing by maintaining a list of recently visited solutions (tabu list) to prevent revisiting and cycling back to previously explored states. (Hurink, et al., 1994)

In this work, I also compared three single-solution-driven algorithms: Steepest Hill Climbing, Iterated Local Search, and Simulated Annealing, on the Traveling Salesman Problem (TSP). The result of the comparison is shown in Figure below.


Best Distance for Each Algorithm
Best Distance
0
100
200
300
400
500
600
700
800
Stochastic Hill Climbing
Steepest Hill Climbing
Iterated Local Search
Tabu Search
Simulated Annealing


Average Distance for Each Algorithm
Average Distance
0
200
400
600
800
Stochastic Hill Climbing
Steepest Hill Climbing
Iterated Local Search
Tabu Search
Simulated Annealing


Standard Deviation for Each Algorithm
Standard Deviation
0
10
20
30
40
50
60
70
80
Stochastic Hill Climbing
Steepest Hill Climbing
Iterated Local Search
Tabu Search
Simulated Annealing

While both Steepest Hill Climbing and Iterated Local Search performed better in terms of best and average distances, I chose Simulated Annealing for further comparison with Genetic Algorithms. The decision was based on Simulated Annealing's ability to balance exploration and exploitation, its adaptability to different problem instances, and its probabilistic mechanism for escaping local optima (Adewole, et al., 2012). Single-driven algorithms are very simple and computationally efficient for small-scale problems. However, their dependency on a single solution trajectory may not allow them to explore the global search space properly. Simulated Annealing does not have this problem, so it is a better candidate to compare with population-based approaches. (Gopalakrishnan Nair & Sooda, 2001).

1.2 Population Driven (Evolutionary) Search Algorithms

Population-driven search algorithms are based on a population of candidate solutions that search in parallel. These algorithms are biologically or socially inspired and operate by mechanisms such as selection, crossover, and mutation. Notable examples include:

- **Genetic Algorithms (GA):** Use evolutionary principles to iteratively evolve a population of solutions. Solutions with better fitness are more likely to survive and produce offspring through crossover and mutation, enabling the algorithm to explore diverse regions of the search space (Adewole, et al., 2012).
- **Particle Swarm Optimization (PSO):** Simulates the social behavior of a flock of birds or school of fish; it updates each particle's trajectory based on its own experience and that of its neighbors (Kennedy & Eberhart, 1995).
- **Ant Colony Optimization (ACO):** Inspired by the foraging behavior of ants, this algorithm builds solutions incrementally, guided by pheromone trails that represent solution quality (Dorigo, et al., 1996).

For the comparison with single solution driven algorithm, I choose Genetic Algorithm. The ability of GA to explore the global search space effectively by population of solutions made it suitable for this problem. Using crossover and mutation operators keeps the population diverse, which prevents stagnation. Population-driven algorithms are very effective for complex problems because they can avoid premature convergence and maintain diversity. However, their parameter tuning is sensitive and computationally expensive.

1.3 Simulated Annealing (SA)

Simulated Annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy, where metals are heated and slowly cooled to achieve a strong crystalline structure. The algorithm works as follows:

1. **Initialization:**
    - Start with an initial solution and set a high initial temperature.

2. **Neighbor Selection:**
    - Generate a neighboring solution by making a small change to the current solution (e.g., swapping two cities in TSP).
3. **Acceptance Criterion:**
    - Calculate the difference in cost (e.g., total distance) between the current solution and the neighbor.
    - If the neighbor is better, accept it. If worse, accept it with a probability proportional to the temperature.
4. **Cooling Schedule:**
    - Gradually reduce the temperature according to a cooling schedule (e.g., exponential or linear decay).
5. **Termination:**
    - Stop when the temperature is sufficiently low or no improvements are observed.

SA's strength lies in its ability to accept worse solutions early in the process, enabling it to escape local minima. As the temperature decreases, the algorithm focuses more on exploitation, converging to an optimal or near-optimal solution.

1.4 Genetic Algorithm (GA)

Genetic Algorithm is a bio-inspired optimization technique that mimics the process of natural selection. The algorithm evolves a population of candidate solutions over multiple generations through the following steps:

1. **Initialization:**
    - Generate an initial population of random solutions (e.g., random tours of cities).
2. **Evaluation:**
    - Calculate the fitness of each solution based on its quality (e.g., inverse of the total distance for TSP).
3. **Selection:**
    - Select parent solutions based on their fitness using methods like tournament selection or roulette wheel.
4. **Crossover:**
    - Combine pairs of parent solutions to produce offspring. For TSP, ordered crossover ensures valid tours.

5. **Mutation:**
    - Introduce small random changes to offspring (e.g., swapping two cities) to maintain diversity.
6. **Replacement:**
    - Replace the worst-performing solutions with new offspring.
7. **Termination:**
    - Stop after a fixed number of generations or when no significant improvements are observed.

GA's population-based approach ensures robust exploration of the search space, making it less prone to getting stuck in local optima. Its adaptability through genetic operators allows fine-tuning for specific problems.

2. Methodology

We will describe in detail the methodology applied in comparison between Simulated Annealing and Genetic Algorithms in solving a TSP problem. We include descriptions of experimental designs, parameter settings, and a performance metric in this section. We will compare algorithms on their effectiveness, robustness and computational efficiency in a fair and systematic way in this section.

To ensure a fair and objective assessment, the following were used:

2.1 Data Preparation

The dataset used, cities.csv, contained the coordinates (x, y) of 50 cities. It was pre-processed to calculate the Euclidean distance between each pair of cities, resulting in a distance matrix. This was the input to both algorithms. As both algorithms used the same data, differences in performance were ascribed solely to characteristics of the algorithms themselves.

2.2 Algorithm Parameters

Both algorithms were initialized with carefully tuned parameters to balance solution quality and computational efficiency. The parameters were chosen based on standard practices from the literature and experimental fine-tuning:

1. **Simulated Annealing (SA):**
    - Hyperparameter tuning was conducted to determine optimal values for:
        - **Initial Temperature**: Governing the acceptance probability of suboptimal solutions.

        - **Cooling Rate**: Controlling the rate at which the algorithm converges.
        - **Stopping Condition (Minimum Temperature)**: Ensuring the algorithm halts after a sufficient search.
    - The fine-tuned parameters were used to run the algorithm consistently across problem sizes.
2. **Genetic Algorithm (GA):**
    - Fixed parameters included:
        - **Population Size**: Number of candidate solutions evaluated in each generation.
        - **Mutation Rate**: Probability of introducing random changes to a solution.
        - **Crossover Rate**: Likelihood of combining two parent solutions.
        - **Number of Generations**: 300 generations were used to evolve the population.
    - **Selection Policy**: Tournament selection was employed to ensure the fittest individuals were chosen for reproduction.
    - **Crossover Policy**: A two-point crossover method combined genetic information from parents.
    - **Mutation Policy**: Inverse mutation was applied to maintain diversity within the population.

2.3 Multiple Runs

Initially, both algorithms were executed for a single run to evaluate their behavior. However, due to the inherent randomness in SA and GA, each algorithm was subsequently run five times for every problem size (10, 20, 30, 40, 50 cities). This ensured robust and reliable performance metrics, capturing the algorithms' average behavior and variability across runs.

2.4 Metrics for Comparison

To evaluate the algorithms' performance, the following metrics were used:

1. **Best Distance:**
    - The shortest route distance discovered by each algorithm across all runs.
    - This metric reflects the ability of the algorithm to find optimal or near-optimal solutions.
2. **Average Distance:**

- The mean route distance across all runs, providing insight into the typical solution quality of the algorithm.
- Solution quality is critical in TSP as the primary objective is to minimize the total route distance.

3. **Computation Time:**
    - The average time (in seconds) taken per run by each algorithm.
    - This metric evaluates the computational efficiency, an essential factor for real-time or resource-constrained applications.
4. **Robustness and Consistency:**
    - **Standard Deviation:** The spread of distances achieved across runs. Lower standard deviations indicate higher consistency and reliability.
    - **Robustness** is vital for applications requiring consistent performance across multiple implementations.

2.5 Implementation Tools

Both algorithms were implemented in Python with the following libraries:

- **NumPy** and **Pandas** for data manipulation and matrix calculations.
- **Matplotlib** for visualizing results and performance trends.

Results and observations were carefully documented to facilitate easy replication and interpretation.

Following this structured methodology, a proper comparison of Simulated Annealing and Genetic Algorithm was possible. Standard parameters, multiple runs, and comprehensive metrics ensured the validity and reliability of the findings. It was on this foundation that in-depth analysis of the strengths, weaknesses, and applicability of these algorithms in solving the TSP was possible.

3. Results

The results of the experiments provide insights into the performance of Simulated Annealing (SA) and Genetic Algorithm (GA) on the Traveling Salesperson Problem (TSP). The following subsections summarize the findings based on key performance metrics.

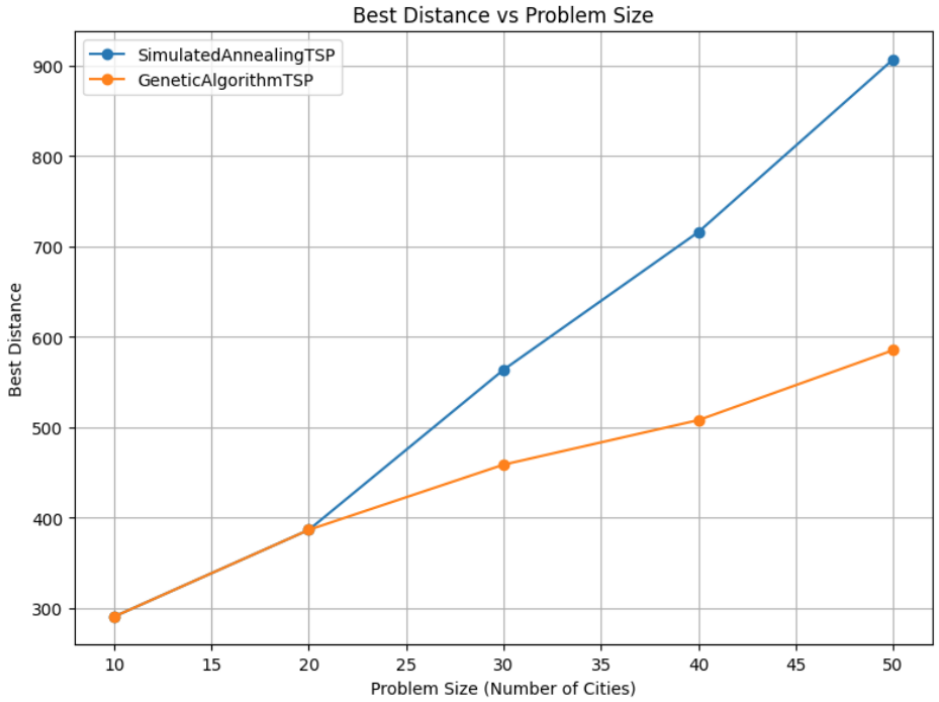
3.1 Best Distance (Quality)

The plot shows how the best distances achieved by SA and the GA change with the number of cities. Clearly, the best distances increase for both algorithms as the problem size grows, reflecting the natural complexity of finding shorter routes in larger problem instances.

A comparison of the two algorithms indicates that at all problem sizes, GA has a shorter best distance than SA. This implies that GA is a more efficient exploration tool in the solution space and can identify near-optimal routes, particularly when the problem size increases. This would imply that the design of GA is better suited to the challenges of larger instances of TSP.

The other significant point of analysis is scalability. While both algorithms increase the best distance with the number of cities, SA increases at a much higher rate. This implies that the performance of SA degrades more rapidly with the size of the problem, hence less scalable than GA. The advantage of GA in solving larger and more complex TSP instances is that it can maintain efficiency and accuracy with increasing complexity.

These are due to the population-based approach: while maintaining different kinds of candidate solutions and utilizing genetic operators, such as crossover and mutation, it allows simultaneous searching of multiple parts of the solution space, thus combining useful properties of a number of solutions and converging eventually to shorter routes. The SA algorithm utilizes the single-solution approach. It iteratively improves one candidate solution. Although this strategy is effective for more modest problems, it severely limits SA's ability to thoroughly explore larger and more complex search spaces.



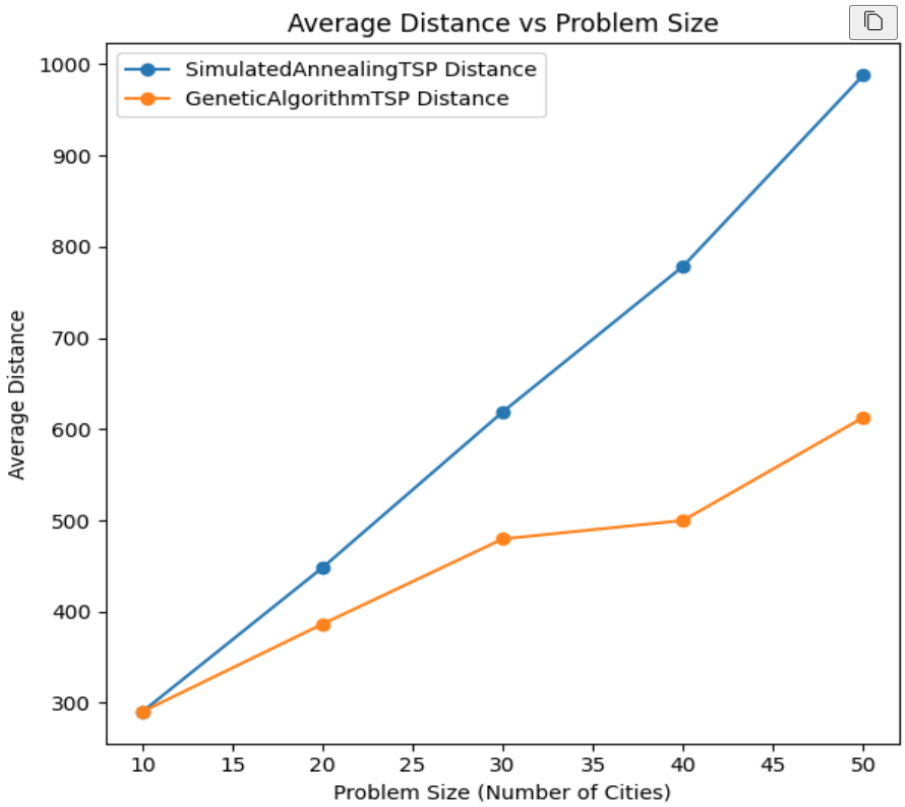
3.2 Average Distance (Quality and Robustness)

Average distance metric was used to measure how good the SA and the GA are in finding optimal or near-optimal solutions. This was achieved by recording the shortest route obtained by each algorithm for different runs and averaging the results for problem sizes between 10 and 50 cities.

Simulated annealing proved competitive in achieving quality solutions, particularly for smaller problem sizes. As the number of cities increases, the performance of simulated annealing begins to decline and plateaus, reflecting the limitations in its capability to adequately search larger, more complex solution spaces.

In contrast, the Genetic Algorithm (GA) outperformed SA in all problem sizes, with shorter average route distances. The population-driven approach of GA allowed it to evaluate diverse solution paths simultaneously, which enhanced its ability to find superior outcomes. As the size of the dataset grew, the gap between the average distances achieved by GA and SA became more pronounced, underscoring GA's scalability and effectiveness in solving larger problems.

Although GA had higher computation times than SA, its tendency to produce better-quality solutions constantly reflects a balance between quality solutions and computation times. On balance, results are convincing regarding the GA's ability and robust performance in addressing complexities of more challenging instances of TSP.



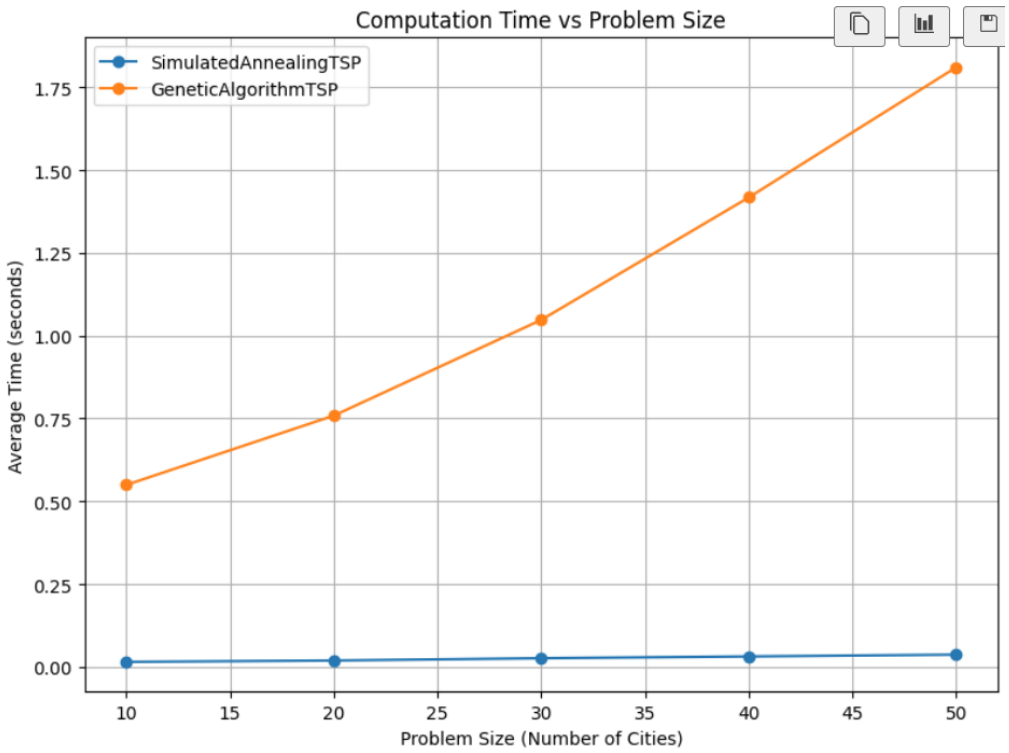
3.3 Computation Time (Efficiency)

The computational efficiency of the SA and GA algorithms is evaluated by the average time used per run. As it was understood, this measure shows how their usage time interacts with solution quality.

Computation times for Simulated Annealing were significantly lower compared to GA. This outcome is expected because SA involves only iterative improvement of a single solution rather than maintaining and evolving a population of candidate solutions. By limiting the scope of its evaluations, SA can converge faster; this makes it a time-efficient alternative if computational resources or time to compute are limited.

On the contrary, Genetic Algorithm took more time for one run. The increased computation is due to the population-based approach of Genetic Algorithm as it evolves a population of candidate solutions through crossover and mutation operations. Although they are highly computationally intensive, such processes allow GA to scan an even larger solution space and provide consistently high-quality solutions.

In scenarios where computational resources are limited or time efficiency is critical, SA is a practical choice. However, the higher computational cost of GA is often justified by its ability to deliver better-quality solutions, particularly for complex and large-scale problems. This trade-off highlights the importance of aligning algorithm selection with the specific priorities and constraints of the problem being addressed.



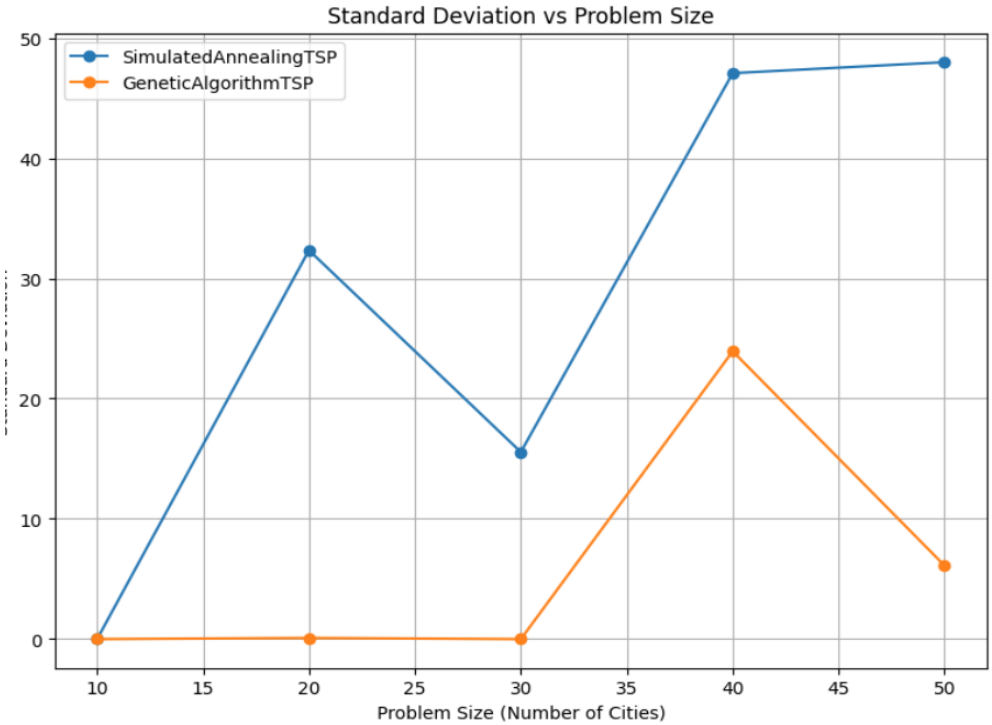
3.4 Standard Deviation (Robustness)

Standard deviation of the route distances obtained at different runs was considered for evaluating the robustness of algorithms. This is a measure that reveals how spread the solutions are and to which extent each algorithm produces solutions of a similar nature.

For SA, the standard deviation tends to increase with the problem size, which reflects more variability in the solutions. This increased variability can be attributed to the stochastic nature of SA, its sensitivity to the initial solution, and the cooling schedule. With the increase in the number of cities, the performance of SA is less predictable and produces high-quality solutions only at times, getting stuck in local optima most of the time.

As opposed to GA, the pattern of standard deviation is more fluctuating. Here, it first increases from 10 to 20 cities, then decreases for 30 cities, increases again for 40 cities, and finally decreases for 50 cities. This fluctuation indicates that the consistency of GA is sensitive to the interactions between problem size, population size, and algorithmic parameters.

Overall, GA tends to have a smaller standard deviation than SA for the larger problem sizes (30 and 50 cities), indicating that GA typically produces more consistent results. A smaller standard deviation entails a greater robustness. For this reason, GA is a more reliable method; it is likely to yield similar quality solutions in a different run. On the other hand, SA's higher variability suggests that running it multiple times with different initial conditions or parameter settings is necessary to obtain a more reliable estimate of its performance..

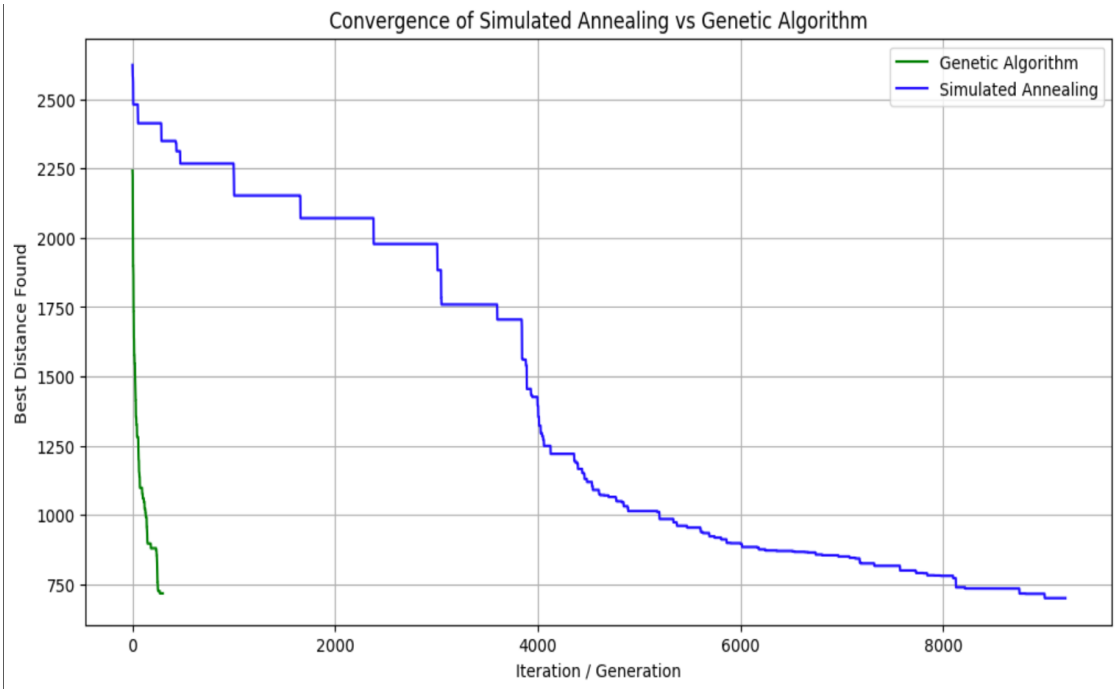


3.5 Convergence Analysis

The convergence analysis highlights the difference between SA and GA search strategies in that the solution improvement over time is actually different.

During the initial stages, the GA improves very rapidly in the solution quality and quickly finds a relatively good solution. The SA starts with a greater initial distance and improves gradually over time. Both algorithms show a general tendency of distance decrease, implying that they are effectively searching for a better solution. However, the GA converges faster, reaching a plateau in solution quality earlier than SA, which continues to make smaller improvements over a longer period. This suggests that the GA tends to exploit existing solutions more effectively, refining them quickly.

That's why GA converges faster, as it involves a population-based search exploring multiple solutions at once, with beneficial traits coming together through crossover. With SA, however, it iteratively improves one solution, and that's the thing: it explores more gradually the solution space since the exploration and exploitation are adapted well through a probabilistic acceptance of worse solutions. Overall, the convergence behavior indicates the strength and trade-offs of each algorithm in terms of exploration-exploitation and solution refinement.



Summary of Findings

- **Solution Quality:** Comparing with Simulated Annealing, GA given a shorter best and average distance which clearly shows the better performance of Genetic Algorithm in finding near optimal solution.
- **Computation Time:** The use of Simulated Annealing was shown to require much less computation time which makes it more favorable 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

4. Recommendations for Algorithm Deployment

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.

4.1 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.

4.2 High Solution Quality Requirements

When achieving the shortest possible route is paramount, Genetic Algorithm is again preferred due to its ability to reach near-optimal solutions in our tests, outperforming the Simulated Annealing in terms of the best and average distances achieved. This makes GA 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.

4.3 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.

4.4 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.

5. 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.

5.1 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.

5.2 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.

6. Conclusion

This project investigated the Traveling Salesperson Problem (TSP) and compared the performance of two metaheuristic algorithms: Simulated Annealing (SA) and Genetic Algorithm (GA). Through detailed implementation, visualization, and performance analysis, we identified the strengths and weaknesses of each algorithm, providing valuable insights into their suitability for solving TSP.

6.1 Key Findings

- **Solution Quality**: We see that GA consistently produced shorter routes compared to SA in terms of best and average distances across the entire range of problem sizes; these results highlight GA's exploratory ability within the solution space, leveraging their population-based approach to combine beneficial traits from multiple solutions.
- **Computational Efficiency**: Simulated Annealing demonstrated significantly faster computation times, particularly for larger problem instances. As a result, SA is more efficient in scenarios where time constraints are crucial.
- **Convergence:** Simulated Annealing was computationally much faster, especially for larger problem instances. Thus, in situations where the execution time is strictly limited, SA becomes more efficient.
- **Robustness:** The GA converged faster and found good solutions quickly but then settled into a plateau in the quality of the solutions obtained. Simulated Annealing exhibited more gradual improvement and suggested that the exploration-exploitation strategy was better balanced.

6.2 Limitations

- The performance of both algorithms seems to be sensitive to hyperparameters. Although the parameters of SA were tuned in this study, further exploration of parameter optimization for both algorithms could enhance results

- The used dataset was specific; hence, extending the analysis to include different instances of TSP with varying characteristics, for example, city distributions and distance metrics, will provide better overall insight into algorithm performance.

6.3 Future Work

- Further investigation of the impact of different parameter settings and cooling schedules on the performance of Simulated Annealing.
- Explore advanced variations of the Genetic Algorithm, such as integrating local search heuristics or adaptive mutation rates, to enhance solution quality.
- Compare SA and GA with other metaheuristic algorithms, like Ant Colony Optimization or Particle Swarm Optimization, to assess their relative effectiveness.
- Apply both algorithms to real-world TSP scenarios to analyze their practical applications and effectiveness in more complex and dynamic environments.

This project offered a comprehensive comparative analysis of Simulated Annealing and Genetic Algorithm for solving the TSP, shedding light on their trade-offs. The findings from this study can guide the selection of appropriate algorithms for optimization tasks across various domains, depending on the specific problem characteristics and computational constraints.

7. References

Adewole, A., Otubamowo, K. & Egunjob, T., 2012. A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem. *nternational Journal of Applied Information Systems,* 4(4), pp. 6-12.

Czech, Z. & Czarnas, P., 2002. *Parallel Simulated Annealing for the Vehicle Routing Problem With Time Windows.* s.l., IEEE.

Dorigo, M., Maniezzo, V. & Colorni, A., 1996. Ant System: Optimization by a colony of cooperating agents.. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society,* Volume 26.

Gopalakrishnan Nair, T. R. & Sooda, K., 2001. Comparison of Genetic Algorithm and Simulated Annealing Technique. *CoRR,* Volume 1001.

Hurink, J., Jurisch, B. & Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum,* 15(4), pp. 205-215.

Kennedy, J. & Eberhart, R., 1995. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks,* Volume 4, pp. 1942-1948.

Rossi, F., Beek, P. v. & Walsh, T., 2006. In: *Handbook of Constraint Programming.* s.l.:Elsevier Science Inc..

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/