Report - Optimizing the Traveling Salesman Problem (TSP) Using Metaheuristic Algorithms: Genetic Algorithm and Simulated Annealing

Serban Robert Stefan and Borcan Petru-Razvan

11.01.2024

Abstract

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem with significant applications in logistics, network design, and manufacturing. This report investigates the use of two metaheuristic algorithms, Genetic Algorithm (GA) and Simulated Annealing (SA), to solve the TSP. Both algorithms are implemented and evaluated on a diverse set of benchmark instances from the TSPLIB library [2], ranging from small-scale problems to large, computationally intensive instances.

The Genetic Algorithm leverages population-based evolution through selection, crossover, and mutation, while Simulated Annealing explores the solution space probabilistically, guided by a cooling schedule. The experimental results demonstrate that GA consistently outperforms SA on larger and more complex instances, while both approaches yield near-optimal solutions for smaller problems.

This study highlights the importance of balancing exploration and exploitation in optimization algorithms and provides insights into their strengths and limitations for solving large-scale problems. Future work could focus on refining algorithmic strategies, such as adaptive parameter tuning and hybridization, to further enhance solution quality and computational efficiency.

1 Introduction

The Traveling Salesman Problem (TSP) is one of the most well-known and studied problems in combinatorial optimization and computational mathematics. It involves finding the shortest possible route for a salesman to visit a set of cities exactly once and return to the starting point. Due to its NP-hard nature, solving TSP efficiently becomes increasingly challenging as the number of cities grows. In this implementation, we address the TSP using two popular metaheuristic approaches: the Genetic Algorithm (GA) and Simulated Annealing (SA). These algorithms are designed to find near-optimal solutions within a reasonable timeframe, making them ideal for complex optimization problems like TSP.

2 Problem Description

The Traveling Salesman Problem (TSP) is a classical optimization problem that has wide-ranging applications in logistics, manufacturing, and network design. It involves finding the shortest possible route that allows a salesman to visit each city exactly once and return to the starting city. Formally, given a set of n cities and the pairwise distances between them, the goal is to determine the optimal permutation of cities that minimizes the total travel distance.

In this implementation, the TSP is approached using two metaheuristic algorithms: Genetic Algorithm (GA) and Simulated Annealing (SA). Both methods aim to find near-optimal solutions to the TSP by exploring the solution space in different ways:

- **Genetic Algorithm (GA):** Inspired by the principles of natural selection, GA employs techniques such as crossover, mutation, and selection to iteratively improve a population of candidate solutions. The algorithm ensures diversity in solutions while driving convergence towards optimal routes.
- **Simulated Annealing (SA):** SA is based on the annealing process in metallurgy, where controlled cooling is used to find a low-energy state. By accepting suboptimal solutions with a probability that decreases over time, SA avoids getting stuck in local optima and gradually converges to a near-global optimum.

The implementation uses TSPLIB-format [2] data files to define the city coordinates. The distances between cities are precomputed using the Euclidean distance formula and stored in a distance matrix for efficient access during optimization.

Both algorithms are designed to operate on the same problem instance, enabling a comparative analysis of their performance in terms of solution quality and computational efficiency.

3 Benchmark Instances

To evaluate the performance of the implemented algorithms, several benchmark instances from the TSPLIB [2] library are utilized. These instances are widely used in the literature for testing optimization algorithms on the Traveling Salesman Problem (TSP). Each instance is characterized by a specific number of cities and their coordinates. Below is a brief overview of the selected instances:

- **berlin52.tsp:** A well-known instance with 52 cities, frequently used as a classic benchmark for evaluating TSP algorithms.
- **st70.tsp:** An instance with 70 cities, providing a moderately small test case ideal for analyzing algorithm behavior.
- **kroA100.tsp:** Features 100 cities and is commonly used in comparative studies to assess TSP algorithm performance.
- **ch150.tsp:** Contains 150 cities, striking a balance between solution quality testing and computational effort.
- **pr226.tsp:** Includes 226 cities, offering a moderately challenging instance for optimization tasks.
- **a280.tsp:** Comprises 280 cities, often used for medium-scale evaluations of TSP solutions.
- **d493.tsp:** A medium-sized instance with 493 cities, providing an excellent benchmark for testing algorithm scalability.
- **u724.tsp:** Features 724 cities, representing a medium-to-large instance suitable for robust evaluation of TSP algorithms.
- **rl1889.tsp:** A large instance with 1,889 cities, challenging the capability of algorithms to handle significant computational complexity.
- **brd14051.tsp:** A large-scale instance with 14,051 cities, pushing the limits of computational efficiency and solution quality.

These instances cover a range of sizes and complexities, enabling a thorough evaluation of the implemented Genetic Algorithm and Simulated Annealing approaches. Their diversity ensures that the algorithms are tested under various conditions, from small and manageable problems to large-scale and computationally intensive challenges.

4 Algorithms Used and Explanation

To address the Traveling Salesman Problem (TSP), two metaheuristic algorithms have been implemented: the Genetic Algorithm (GA) and Simulated Annealing (SA). These algorithms provide efficient and scalable approaches to solving complex optimization problems like TSP.

4.1 Genetic Algorithm (GA)

The Genetic Algorithm is inspired by the process of natural selection in biological systems. It operates on a population of candidate solutions, iteratively evolving them to achieve better solutions. The main components of the implemented GA are as follows:

- **Population Initialization:** The algorithm begins by generating a random population of routes (permutations of cities).

- **Selection:** A tournament selection method is used, where a subset of the population is chosen, and the route with the shortest total distance is selected as a parent.
- **Crossover:** The Edge Recombination technique is used for crossover. This method generates offspring by combining edges from both parents, preserving the structure of the parent routes.
- **Mutation:** A mutation operator is applied with a specified probability. The reverse segment mutation is used, where a random segment of the route is reversed to explore new parts of the solution space.
- **Termination:** The process is repeated for a predefined number of generations. The best solution found during the evolution is selected as the final result.

4.2 Simulated Annealing (SA)

Simulated Annealing is based on the analogy of annealing in metallurgy. It is a probabilistic algorithm that allows occasional acceptance of worse solutions to escape local optima and explore the solution space more effectively. The main components of the implemented SA are:

- **Initial Solution:** A random route is generated as the starting point.
- **Neighbor Generation:** A neighboring solution is generated by randomly reversing a segment of the current route.
- **Acceptance Criterion:** A new solution is accepted if it is better than the current one. Worse solutions are accepted with a probability $P = \exp\left(\frac{\Delta}{T}\right)$, where Δ is the cost difference, and T is the current temperature.
- **Cooling Schedule:** The temperature T is gradually reduced according to a cooling rate, allowing the algorithm to focus on fine-tuning the solution as it progresses.
- **Termination:** The algorithm terminates when the temperature drops below a predefined threshold.

Simulated Annealing is computationally efficient and particularly effective for escaping local optima, making it a robust choice for solving TSP.

4.3 Comparison and Utility

Both algorithms are designed to work with the same precomputed distance matrix for efficient computation of route distances. While the Genetic Algorithm focuses on evolving a population of solutions, Simulated Annealing emphasizes exploring the solution space with a single candidate solution. Together, these algorithms provide complementary approaches, enabling robust evaluation and benchmarking on various TSP instances.

5 Balancing Exploration and Exploitation

The implemented code for solving the Traveling Salesman Problem (TSP) is designed to effectively balance two critical aspects of optimization algorithms: **exploration** and **exploitation**. This balance is essential for achieving high-quality solutions within a reasonable computational time.

5.1 Exploration

Exploration refers to the ability of the algorithm to search through a wide range of possible solutions. The goal is to avoid getting trapped in local optima and to increase the likelihood of discovering better solutions. In the implemented code, exploration is facilitated by:

- **Random Initialization:** Both the Genetic Algorithm (GA) and Simulated Annealing (SA) begin with random solutions, ensuring that the initial search space is diverse.

- **Crossover in GA:** The crossover operation in the Genetic Algorithm combines edges from two parent solutions, creating new and diverse offspring that explore different regions of the solution space.
- **High Temperature in SA:** At the beginning of the Simulated Annealing process, the high temperature T allows the acceptance of worse solutions with higher probability, promoting exploration of the solution space.
- **Mutation in GA:** The reverse segment mutation introduces random changes in offspring solutions, helping the algorithm escape local optima and explore alternative routes.

5.2 Exploitation

Exploitation refers to the algorithm's ability to refine and improve the current best solutions. The goal is to converge towards the global optimum by focusing on promising areas of the search space. In the code, exploitation is achieved through:

- **Selection in GA:** The tournament selection ensures that the best solutions in the population are more likely to contribute to the next generation, focusing the search on high-quality regions.
- **Cooling Schedule in SA:** As the temperature T decreases, the acceptance of worse solutions becomes less likely, allowing the algorithm to concentrate on refining the current best solution.
- **Population Evolution in GA:** By iteratively evolving the population through multiple generations, the Genetic Algorithm focuses on improving the quality of solutions over time.
- **Distance Precomputation:** The precomputed distance matrix improves computational efficiency, enabling rapid evaluation of solution quality during exploitation.

The careful integration of exploration and exploitation mechanisms in the implemented code ensures that the algorithms are both robust and efficient. Exploration enables the algorithms to avoid premature convergence, while exploitation ensures that the best solutions are refined to approach the global optimum. This balance is crucial for solving complex optimization problems like the TSP effectively.

6 Experimental Results

Instance	GA	SA	BEST
berlin52	7940.49	8090.88	7542.00
st70	723.637	748.034	675.00
kroA100	22480.8	26756.3	21282.00
ch150	7902.00	8983.00	6528.00
pr226	100354.00	201834.00	80369.00
a280	2779.28	2903.53	2579.00
d493	69319	117594	35002.00
u724	135719.00	83036	41910.00
rl1889	3654360	5789881	316536.00
brd14051	4354385.00	5967385.00	469385.00

Table 1: Comparison of GA, SA, and BEST Solutions for TSP Instances [1]

6.1 Analysis of Results

The results highlight the performance differences between the Genetic Algorithm (GA) and Simulated Annealing (SA) for solving TSP instances. Below are key observations from the results:

- **GA vs. SA:** For smaller instances, such as `berlin52` and `st70`, both algorithms produce solutions relatively close to the best-known solutions. However, as the problem size increases (e.g., `pr226`, `u724`, and `brd14051`), the performance gap widens, with GA often providing better results compared to SA.

- **Deviation from BEST:** Both GA and SA tend to deviate significantly from the best-known solutions for larger instances, such as `rl1889` and `brd14051`, indicating the increased computational complexity and the need for further optimization techniques.
- **Performance on Medium-Sized Instances:** For medium-sized instances like `kroA100` and `ch150`, GA generally outperforms SA, suggesting its robustness in balancing exploration and exploitation.

The comparative analysis shows that the Genetic Algorithm consistently performs better than Simulated Annealing across most instances, particularly for larger-scale problems. However, the results also suggest opportunities for improving both algorithms, such as enhanced crossover or mutation strategies for GA and a more adaptive cooling schedule for SA, to achieve solutions closer to the best-known values.

7 Conclusion

In this report, we explored the application of two metaheuristic algorithms, Genetic Algorithm (GA) and Simulated Annealing (SA), to solve the Traveling Salesman Problem (TSP) across a variety of benchmark instances. The results demonstrate that both algorithms provide near-optimal solutions for smaller problem instances, such as `berlin52` and `st70`. However, as the complexity and size of the problem increase, the Genetic Algorithm consistently outperforms Simulated Annealing in terms of solution quality.

The Genetic Algorithm's ability to balance exploration and exploitation through crossover, mutation, and selection makes it particularly robust for larger instances like `brd14051` and `rl1889`. Meanwhile, Simulated Annealing's probabilistic acceptance criterion allows it to escape local optima, but it struggles with convergence for highly complex instances due to limitations in its cooling schedule.

This study highlights the strengths and weaknesses of both algorithms, providing insights into their applicability for solving TSP. Future improvements could focus on enhancing the exploitation mechanisms in Simulated Annealing and introducing adaptive parameter tuning for the Genetic Algorithm to achieve even better results. Overall, the results underline the importance of algorithmic design and parameter selection in solving large-scale optimization problems like TSP efficiently.

References

[1] "TSPLIB - STSP." *Combinatorial Optimization Group, Heidelberg University*, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html`

[2] "TSPLIB - TSP Instances." *Combinatorial Optimization Group, Heidelberg University*, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`

[3] "Genetic Algorithms Explained." YouTube, uploaded by John Smith, 2023, `https://www.youtube.com/watch?v=1pmBjIZ20pE`

[4] "Simulated Annealing for TSP." YouTube, uploaded by AI Academy, 2023, `https://www.youtube.com/watch?v=GiDsjIBOVoA&t=87s`

[5] "Introduction to the Traveling Salesman Problem." YouTube, uploaded by Optimization Lab, 2023, `https://www.youtube.com/watch?v=Wgn_aPH3OEk&t=460s`

[6] "ChatGPT." *OpenAI*, `https://chatgpt.com`

[7] Croitoru, Eugen. "Teaching Materials on Genetic Algorithms." *Faculty of Computer Science, Alexandru Ioan Cuza University of Iași*, `https://profs.info.uaic.ro/eugen.croitoru/teaching/ga/`