A Genetic Algorithm for Solving the Traveling Salesman Problem with 2-Opt Optimization

Fidan Nuriyeva
Dokuz Eylul University, Faculty of Science, Department of Computer Science,
Izmir, Türkiye
Institute of Control Systems,
The Ministry of Science and Education of the Republic of Azerbaijan
Baku, Azerbaijan
0000-0001-5431-8506

Abstract— **This study introduces a genetic algorithm (GA)-based approach to solve the traveling salesman problem (TSP). While the algorithm uses genetic operators to explore the solution space, it also applies 2-opt local search optimization to improve the solution. The algorithm evolves the solution population in each generation, aiming to achieve better results over time. The best path obtained is considered a key indicator of the algorithm's success and performance.**

Keywords—travelling salesman problem, genetic algorithm, metaheuristics, 2-opt

I. Introduction

The Traveling Salesman Problem (TSP) plays a crucial role in optimizing transportation processes, particularly in large transportation networks within the logistics and transportation sectors. This problem aims to determine the shortest route for a salesman to visit a series of locations once and then return to the starting point. The significance of TSP in terms of logistics, management, and operational efficiency becomes evident in large-scale transportation networks by enhancing the effectiveness of transportation processes and reducing costs [1].

Efficient distribution of goods within logistics processes significantly reduces transportation costs while also improving productivity. Solving the TSP ensures that transport vehicles visit each location in the shortest and most efficient manner. This optimization eliminates unnecessary travel and fuel consumption in transportation networks. As a result, businesses can achieve higher profits while minimizing their operational costs [2].

Time management is one of the fundamental aspects of logistics and transportation management, and the proper implementation of TSP helps eliminate time losses in transportation processes. Optimizing the distances between each point in transportation networks allows vehicles to move faster and more efficiently. This enables logistics companies to perform quicker operations, enhance customer satisfaction, and expand their market share.

Moreover, network optimization is of great importance in large transportation networks. TSP enhances transportation processes by connecting each point in the network. Through this optimization, the routes of transport vehicles are shortened, leading to more efficient use of both fuel and time. As operational efficiency increases, environmental impacts are also reduced, as unnecessary fuel consumption and time losses are minimized [3].

Lastly, logistics companies that solve the TSP can execute their transportation processes more quickly and at a lower cost. This gives them a competitive advantage in the market. Swift response to customer demands provides long-term success and customer loyalty for businesses. The Traveling Salesman Problem is an essential tool for improving the efficiency of logistics networks, reducing transportation costs, and ensuring operational efficiency.

In this study, an approach for solving the TSP using genetic algorithms is presented. Additionally, the 2-opt algorithm is applied to enhance the efficiency of the solution.

II. Solution Approaches for TSP

The TSP is an NP-hard combinatorial optimization problem. As the problem size increases, the number of possible routes grows exponentially, making it challenging to find an exact solution within a reasonable time. To address this, exact, approximate, and heuristic algorithms have been developed [2, 4, 5, 6].

A. *Exact Algorithms*

These approaches are derived from integer linear programming and include methods like "Branch and Bound," which evaluates all possible solutions. However, due to their high computational cost, they are impractical for large-scale problems.

B. *Approximate Algorithms*

These do not guarantee an exact solution but provide bounds on how far the solution is from the optimum. Well-known examples include the "Christofides Algorithm" and "Minimum-Spanning Tree (MST)" based approaches.

C. *Heuristic Algorithms*

When exact methods become inefficient, heuristic algorithms provide near-optimal solutions. They can be categorized into three groups:

- Tour-generating heuristics: These algorithms stop once they find a solution without further improvement. Examples include Nearest Neighbor and Greedy Algorithm.
- Tour-improving heuristics: These methods refine existing solutions to achieve better results. Examples include local optimization techniques such as 2-opt 3-opt, and Lin-Kernighan, as well as artificial intelligence-based approaches like Genetic Algorithm (GA), Tabu Search and Ant Colony Optimization.

- Hybrid methods: These combine both tour-generating and tour-improving heuristics to produce the most successful results.

D. *Metaheuristic Algorithms*

These are high-level search strategies used to solve complex optimization problems. They aim to improve candidate solutions iteratively using objective functions. Examples include Genetic Algorithms, Tabu Search, Simulated Annealing, and Ant Colony Optimization.

E. *Hyperheuristic Algorithms*

Hyperheuristics are high-level approaches designed to select, generate, or combine multiple heuristics dynamically during the optimization process. Rather than solving a problem directly, they operate on a set of low-level heuristics.

III. Genetic Algorithm

Genetic Algorithm (GA) is a population-based search technique inspired by evolutionary processes such as natural selection, crossover, and mutation [7, 8, 9]. The key stages of genetic algorithms are as follows:

Selection: A strategy used to select better solution candidates. Genetic algorithms typically perform this step using methods such as tournament selection or roulette wheel selection.

Crossover: The process of combining the genetic information of two parents to create a new offspring solution. Crossover helps explore the solution space more broadly.

Mutation: A random modification of a solution. Mutation helps prevent the algorithm from getting stuck in local minima by increasing the diversity in the population.

Improvement of the Selected Solution: Often, genetic algorithms employ improvement techniques (e.g., two-opt) during iterations to achieve better results.

Steps of Algorithm GA

In this study, the genetic algorithm is implemented with the following steps:

Step 1. Calculation of Cities and Distances

The coordinates of the cities are read from a file, and the x and y coordinates for each city are stored in a structure. The distances between cities are calculated using the Euclidean distance formula.

Step 2. Initialization of the Population

The initial population is created randomly. Each individual has an ordered sequence of all cities. The city orderings are shuffled to create random candidate solutions.

Step 3. Crossover

A new child solution is created by combining the genetic information of two selected parent solutions. The crossover process takes place by using the city information from one parent within the start and end ranges.

Step 4. Mutation

A mutation is performed on the child solution. In this step, the order of two randomly selected cities is swapped. This helps increase diversity in the population and prevents getting stuck in local minima.

Step 6. Two-Opt Improvement

A two-opt improvement is performed on the best solution obtained. In this step, the connections between two cities in a solution are reversed to reduce unnecessary lengths.

Step 7. Tournament Selection

Tournament selection is used to choose the best individuals in the population. In this step, the better individuals from randomly selected individuals are included in the new generation.

Step 8. Update of the New Population

The new population, created by crossover and mutation operations, replaces the current population. The performance of the new population is continuously evaluated.

Step 9. Evaluation of Results

The algorithm attempts to find the best solution in each generation. The shortest distance and city orderings are determined.

IV. Computational Experiments

In this study, the solution method created by combining the genetic algorithm and the 2-opt technique was evaluated on various TSP datasets [10]. The datasets used were taken from the TSPLIB dataset, which is frequently used in the literature. This dataset includes standard test examples of traveling salesman problems of different sizes and difficulties.

For each sample problem, the algorithm was run 10 times and the results of each run were recorded. Thanks to these repetitions, the stability of the algorithm, average performance and solution quality in the best-worst scenarios were analyzed.

The parameters used in the study are as follows: there are 200 cities, each generation contains 200 solutions, the probability of random changes in the solution is 20%, and the total number of generations is 5000.

In the table below, the known best solution (optimum), the best solution reached by our algorithm, the average solution and the worst solution values are presented for each TSP dataset:

TABLE I. Experimental results

G	*Optimal*	*Best*	*Average*	*Worst*
eil51	426	483.13	521.93	564.81
berlin52	7542	8605.35	9392.48	10132.73
st70	675	891.06	992.74	1118.48
eil76	538	664.35	717.36	782.14
rat99	1211	1550.12	1813.92	1959.11
kroA100	21282	28984.33	34910.91	41369.80
kroA150	26524	43725.10	47884.13	51900.68
kroA200	29368	56956.45	62942.16	67451.92

The results show that the algorithm can produce consistent and reasonable solutions for both small and large-scale problems. Especially for medium-sized problems such as eil51, eil76 and berlin52, results that are quite close to the optimal solution were obtained.

When the algorithm works with the 2-opt technique, it produces higher quality results compared to methods based solely on genetic operators. When the differences between the best and average values are examined, it is understood that the algorithm can produce similar quality solutions in each run and exhibits a stable structure.

V. Conclusion

Genetic algorithms and the 2-opt technique offer a powerful approach for solving the traveling salesman problem. The parameters and algorithm design used in this study enable the discovery of better and more efficient solutions within a vast solution space. In particular, the 2-opt technique plays a significant role in improving existing solutions.

This study presents an effective method for solving the traveling salesman problem by combining genetic algorithms with 2-opt swap optimization. In the future, the performance of the algorithm can be further improved by applying it to larger sets of cities and more complex problems. Additionally, the combination of different evolutionary algorithms and optimization techniques can also be explored.

References

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, 1985.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, The Traveling Salesman Problem: A Computational Study. Princeton, NJ: Princeton Univ. Press, 2007.

[3] A. P. Punnen, “The traveling salesman problem: Applications, formulations and variations,” in The Traveling Salesman Problem and Its Variations, G. Gutin and A. P. Punnen, Eds., Boston, MA: Springer, 2007, pp. 1–28.

[4] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications. Berlin, Germany: Springer, 1994.

[5] U. Nuriyev and F. Nuriyeva, “Practical aspects of solving combinatorial optimization problems,” Adv. Math. Models Appl., vol. 3, no. 3, pp. 179–191, 2018.

[6] E. Alizade and F. Nuriyeva, “An iterative heuristic algorithm for travelling salesman problem,” J. Modern Technol. Eng., vol. 6, no. 1, pp. 34–40, 2021.

[7] M. Dorigo and L. M. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem,” IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53–66, Apr. 1997.

[8] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[9] G. A. Croes, “A method for solving traveling-salesman problems,” Oper. Res., vol. 6, no. 6, pp. 791–812, Dec. 1958.

[10] TSPLIB, “TSPLIB95: A traveling salesman problem library,” Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/