NTU Journal of Pure Sciences (2025) 4 (1) : 39-47
DOI: https://doi.org/10.56286/ntujps.v4i1

IRAQI Academic Scientific Journals

JPS NTU

ISSN: 2789-1089 EISSN: 2789-1097
NTU Journal of Pure Sciences
Available online at: https://journals.ntu.edu.iq/index.php/NTU-JPS/index

Optimizing Path of The Travelling Salesman Problem Through Modified Genetic Algorithms

Osama A. Qasim
Northern Technical University, Nineveh Technical Institute of Administration

Article Informations

Received: 27-02- 2025,
Accepted: 20-03-2025,
Published online: 31-03-2023

Corresponding author:
Name: Osama A. Qasim
Affiliation : Northern Technical University
Email: moh_sami@ntu.edu.iq

Key Words:
Modify,
Travelling Salesman Problem,
Genetic Algorithm.

A B S T R A C T

The Traveling Salesman Problem (TSP) stands as one of the earliest and most pervasive optimization challenges, aiming to streamline the salesman's travel route, ensuring efficiency and avoiding redundancy. With an extensive number of cities to visit and the requirement to find the shortest route, solving the TSP becomes computationally daunting. To tackle this, various approaches are explored, including the genetic algorithm. This study employs a modified genetic algorithm, inspired by biological processes such as population dynamics, crossover, mutation, and natural selection. Introducing a novel intersection approach amalgamating sliding flipping and swapping methods, the study investigates towns with populations ranging from (10, 50) to (150, 200) across two scenarios. In the first scenario, experiments were conducted with a fixed iterations (10000) and a population is (150), resulting in execution times of 20.00 s, 55.14 s, 37.11 s, and 120.14 s when choose the cities (100, 200, 10, 20), respectively. In the second scenario, the population sizes of 50 or 100, and iteration numbers of 1000 or 500 were explored. The times needed for solution determination were 01.20 s, 02.30 s, 02.74 s, and 17.15 s for cities (10, 20, 100, 200), in accordance. Notably, the study's outcomes indicated that there is a proportional relationship between iteration, population, and the number of cities. When integrating the three techniques, a shorter and faster path is achieved compared to the standard genetic algorithm outcomes

Introduction

It has become necessary to use graphic computing in many industries due to the exponential growth of big data technology in order to expedite the identification of quick solutions that reduce effort and time. The path between two points is very important in reducing the distance travelled, the task completion time and the cost [1]. In this study, the problem of a traveling salesman was chosen and how to find its solution. The problem of the traveling salesman can be described as the seller must visit a group of randomly selected cities in a graph. Where the sales representative visits all of the cities once without repeating, and stops visiting at the point from which he started [2]. The traveling salesman problem has become one of the most well-known and extensively explored combinatorial optimization problems. It is extremely tough to solve either optimally or semi-optimally. A traveling salesman problem is used to identify the solution, making it a standard in numerous algorithm designs [3]. A genetic algorithm founded on Darwin's principle of "survival of the fittest" is provided. It supports the notion of evolution and natural selection, which is applied to a wide range of challenges. Optimization challenges, such as the traveling salesman issue [4]. The genetic algorithm is one of the earliest and most widely used methods of randomized efficiency, founded on natural phenomena influenced by genetics. The genetic algorithm employs a natural selection strategy to identify the individuals most suited for reproduction.

This approach comprised of five steps: initializing the population, selecting individuals, crossover, evaluating fitness, and introducing mutations. The basic condition is that initial generation (parents) exhibits physical fitness, offspring (subsequent generation) are likely to inherit these traits, enhancing their chances of survival. This cycle continues until the optimal generation is reached [5]. Many scientists and researchers have utilized the genetic algorithm, recognized considered as one of the most efficient algorithms for optimization searches. Given the significance of durability and reliability in the global landscape, its methodologies have gained widespread popularity [6]. Employ the genetic algorithm to find the finest navigation satellites. Take full advantage of the GPS system, and hybrid-GA approach to the travelling salesman problem [7], [8].

Genetic Algorithm

Genetic Algorithm (GA) is heuristic search optimization algorithm, belonging to the categories that make up the class Evolutionary algorithms which replicate natural evolution processes. GA is a community-based engine optimization technique that develops a new population using three core operations: selection, crossover, and mutation. Widely applied across various fields, including computer networks, the genetic algorithm stands out for its versatility and effectiveness [9], speech recognition [10], picture processing [11], and computer systems engineering [12]. The subsequent example illustrates a basic genetic algorithm:

Step 1: Generate random population of possible solutions (starting populations).
Step 2: Calculate the f(x) fitness statistic for every population member.
Step 3: To develop an entirely novel population, resume the first three steps until completed.
Step 4: Choose two people from the present generations to mate with.
Step 5: Use a particular crossing ratio to generate offspring.
Step 6: Use a particular mutation ratio.
Step 7: Repeat the previous steps until every one of the criteria is met.

Genetic Operators

Genetic Algorithm consists of the following essential steps:

• Mutation
• Evaluation
• Cross over

We aim to elucidate the primary objective of the algorithm: finding a short-closed path by integrating multiple techniques in the mixed approach, namely flipping, swapping, and sliding. Initially, the flip method is employed to generate a new path for the next generation. Subsequently, the swap mechanism is used on this new generation route. Ultimately, slip technology is used to optimize the path further, culminating in the final generation, which combines all three technologies. After adopting this method within the program and doing comparison analysis, it was discovered that the algorithm's efficacy increased greatly with regard to of execution effectiveness, length travelled, and the discovery of an ideal path without duplication across the many cases examined.

Genetic Algorithm Method Flow

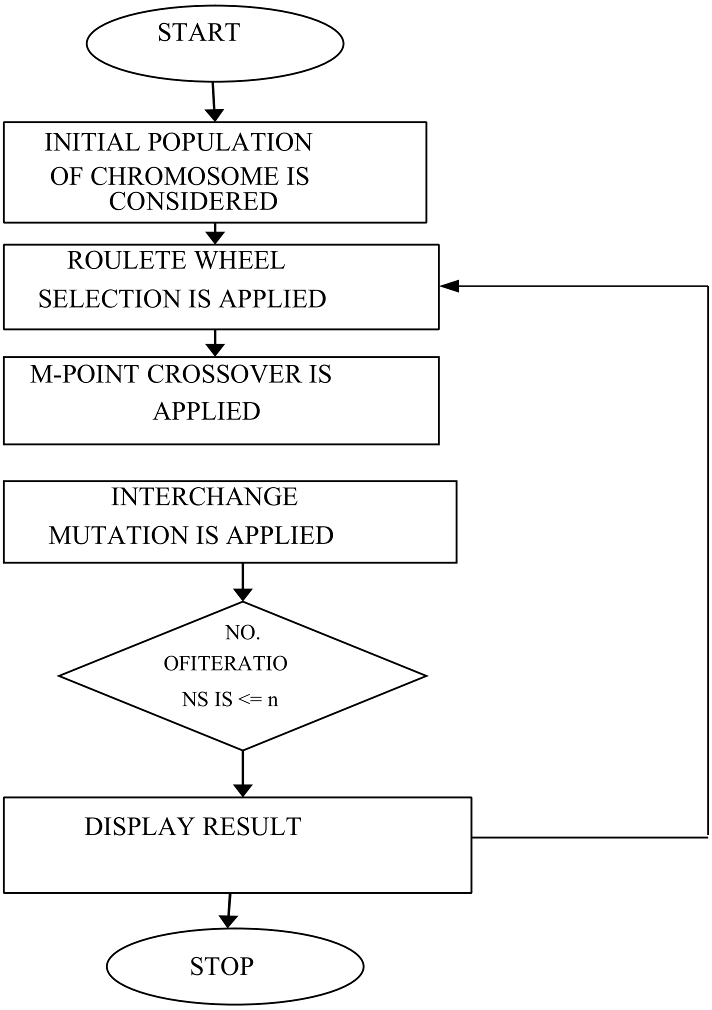

Figure 1. Method flow of genetic algorithm

Travelling Salesman Problem

Traveling salesman problem is a renowned challenge in the realms of operations research and computer science. Which seeks to calculate the least expensive Hamiltonian cycle in a number of cities. This problem can be described as follows: Give values for a number of cities and calculate the distances among them. Starting with the first individual city, then visiting the rest of the cities and returning to the starting point, where the seller must visit all the remaining cities only once, without visiting any city again, choosing the shortest (cost) route to travel. The TSP problem has been widely studied by a lot of researchers, and for this, several methods have been proposed to solve it city precisely once and then return to the starting point [13]. To solve this problem, we must convert it into mathematical equations and represent these equations through a graph showing the number of cities and the proposed paths between cities, and by applying these equations using the genetic algorithm, we will get the best and shortest suggested path and also save time to reach the goal by visiting all cities and return to the starting point as well we can we can describe and represent it according to the following definition [14].

Initial Population

The foundational step in the genetic algorithm is the creation of communities. At this stage, random chromosomes are generated, with their number proportional to the population size, and each chromosome is represented uniquely based on the nature of the challenge. The fundamental generation is the optimal point in addressing a complicated issue. The first set contains of chromosome created at random during development MATLAB software's random generation tool or selected at a predetermined quantity for each experiment. This

study adopted a fixed number and conducted research across multiple cities. Enhancing the base generation serves as a crucial step in tackling the complexity of the problem [15].
Set of chromosomes used are as follows:

Chromosome 1:3 1 4 6 8 7 6 4 8 7 6 3 2 6 8 9 4 7 1 4
Chromosome 2:7 8 4 5 9 4 2 4 6 8 7 7 6 5 2 9 5 4 8 5
Chromosome 3:9 6 5 7 9 2 2 4 8 9 1 6 3 2 9 6 8 1 5 4
Chromosome 4:4 6 8 3 1 8 6 1 7 3 2 7 9 1 6 9 5 8 3 2
Chromosome 5:6 7 8 3 5 8 5 8 1 6 4 1 7 7 5 7 0 4 2 3
Chromosome 6:3 6 5 8 9 8 5 8 1 2 4 3 8 0 8 6 3 6 0 2
Chromosome 7:4 7 8 4 3 5 7 9 8 0 8 4 9 4 8 2 5 0 6 9
Chromosome 8:5 7 3 9 5 8 8 0 4 3 0 7 4 0 7 4 8 5 9 5
Chromosome 9:2 4 3 6 7 4 9 4 8 0 6 8 9 4 7 8 0 0 3 8
Chromosome 10:5 6 8 8 4 3 7 8 0 9 7 7 0 4 6 3 5 2 4 8

Fitness Value

The criteria used to select the best chromosome from among the different possible children is the fitness function. The distance between the cities serves as the TSP chromosome's fitness criterion.

Selection

Selection is necessary to pick the chromosome with the lowest fitness value among all potential chromosomes. In this study, the wheel of roulette selection method was applied by using a sorting approach based on fitness value.

Hybrid Methods Crossover (Sliding, Swapping, And Flipping)

Hybrid crossover methods incorporating sliding, swapping, and flipping are proposed in this paper. A novel intersection approach is introduced, combining different technique: flip, swap, and sliding. Initially, flip method is applied to generate the next generation, followed by the swapping method on the resulting generation. The sliding technique is employed to create the last generation, which amalgamates all three technologies. Upon implementing this method into the program and conducting comparative analysis, it was observed that the algorithm's efficiency significantly improved regarding duration and distance covered as well as the determination of the optimal route devoid of repetition across various scenarios tested. The following code demonstrates this approach.

Mutation

Mutation is a simple technique that alters or substitutes a specific element within an individual as a result of a mating process, with the particular value being selected randomly. It is typically employed to enhance the genetic traits of individuals within a population. The mutation process varies depending on how the chromosome is represented. When the chromosome is displayed in binary form, consisting of a set of (0, 1), mutation involves replacing a one with a zero or vice versa—a strategy known as inversion. However, when chromosomes are represented as real or integer values, mutation entails adding a specified number to the value at the targeted position [16].

Experimental Result

This section presents, discusses, and evaluates the results of the dissertation, in accordance with the implementation outlined in the preceding section. We delve into the intricacies of applying the application of the genetic algorithm is spread throughout an arbitrary number of cities. First, a set population size and a predefined number of iterations are selected, producing a range of results (Tabulated). Subsequently, we conduct further experiments on the same cities utilized in the initial trial, employing increased iterations and a larger population size, with all data being documented in Tables 1 and 2. The algorithm determines the ideal generation crossover and, consequently, the intended new path by integrating three hybrid approaches: flip, swap, and slide. Following the results presentation, we compare the algorithm's performance in terms of duration, distance traveled, and finding the best path without repeating steps across several circumstances. Finding the most effective route in the least amount of time and without duplication is still the key goal.

Experimental Setup

The major goal of this part was evaluating how well the suggested genetic algorithm performs by applying different parameter values to each instance and comparing the results [17]. In the first test, a sample of ten cities was chosen, and the genetic algorithm was run under two different circumstances. The first scenario had a population size of 150 and 100 iterations, whereas the second scenario kept the population size at 150 but increased the number of iterations to 10,000. In the subsequent test, 50 communities were picked, and the algorithm was assessed under two different circumstances; the first comprised 10,000 iterations and a population of 150. In the third test, 100 cities were chosen. The test was performed in two cases. The first scenario included determining the number of iterations (100) and population size (150). In the second scenario, the population size (150) and number of iterations (10,000) were selected. In the fourth test, 200 cities were chosen, and the test was done on two situations. In the first scenario, the number of iterations (1000) and population (150) were selected. In the second scenario, the population number of 150 and the number of iterations of 10,000 were selected. After the results were revealed, the studies were repeated in the preceding cities (10, 50, 100, 200). However, recurrence values (100, 10,000, 5,000) and population size (50,100,100, and 50) were selected, and the results were placed in a table and compared with prior results, and the best instances.

This is the graphical depiction of the optimal solution history with 10,000 iterations and a population size of 150 cities chosen at random: This graphic illustrates how the algorithm's best answer changed during the course of the iterations.

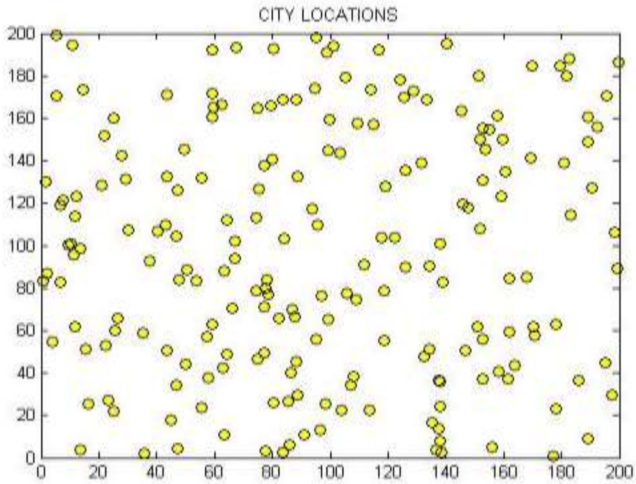

Figure. 2. Depicts the city locations of (200) randomly generated cities.

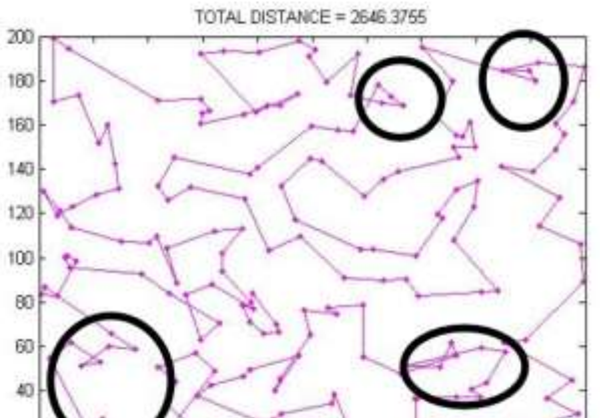

Figure. 3. Total Distance (200) City.

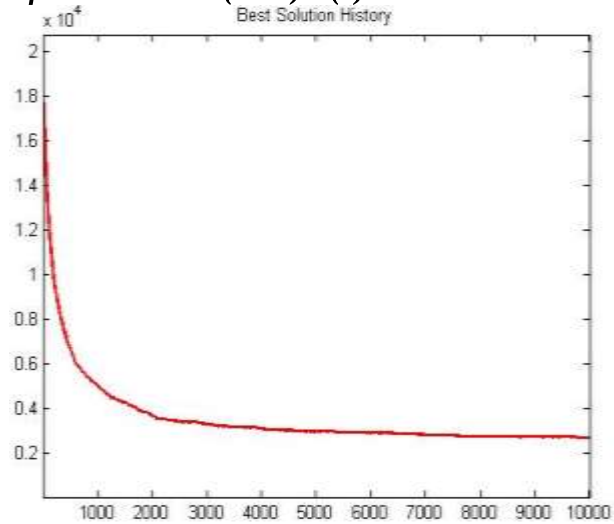

Figure. 4. Depicts the best solution history for (200) cities with (1000) iterations.

It was obsereved from Table 1, that now is the perfect moment to choose the best route (17.97 s), although this is not the optimum time. In the rerun of the experiment with (200) cities, a population size of (200) and (100) iterations, in order to proof affects the findings was applied. Then, these findings were compared with the optimum ones to show how upgrading the genetic algorithm have improved the accuracy. This comparison will provide insight into the modified algorithm's performance in selecting the ideal route and time when compared to traditional genetic algorithms.

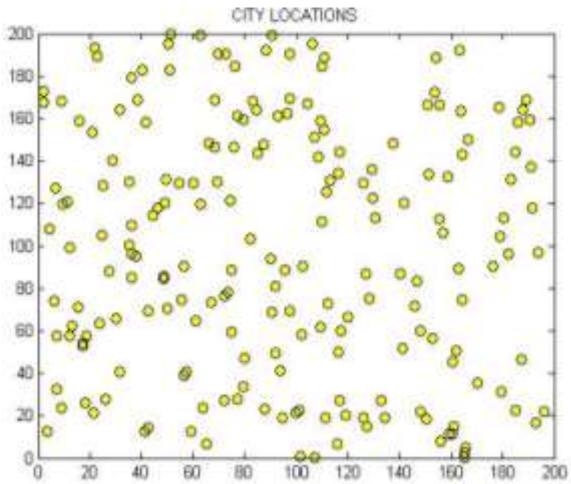

Figure. 5. The coordinates provided represent the spatial positions of the cities, delineated by their respective x and y coordinates

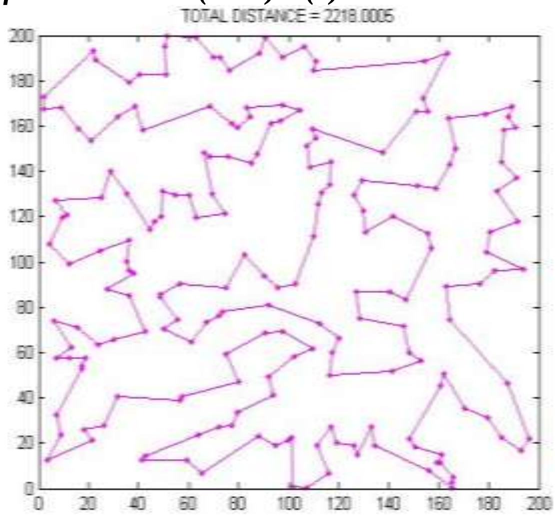

Figure. 6. Population size (100), total distance for 200 cities.

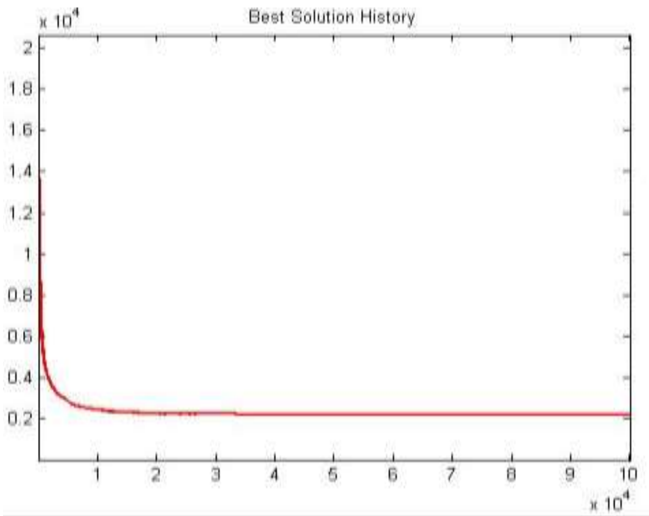

Figure 7. Depicts the optimal solution history for (200) cities with (10000) iterations.

Table 1. Results of the Algorithm using two operator Time and Distance using Conventional Genetic Algorithm

Cities	Iterations	Population	Distance	Time to complete find the best path
10	10000	150	15.7948	41.11(s)
10	100	150	12.4193	01.71(s)
50	10000	150	146.1050	55.14(s)
50	100	150	119.3410	02.73(s)
100	1000	150	196.43207	11.65(s)
100	10000	150	404.1325	75.11(s)
200	10000	150	6167.219	120.14(s)
200	1000	150	1241.7553	17.97(s)

To address the problem of the traveling salesman using a genetic algorithm, the experiment varied the number of iterations for different group sizes of cities (10, 50, 100, and 200). Initially, a common number of iterations, 1000, was selected for testing.

The results showed that in a small number of cities (1050) the time to find the best route will be optimal, and in large cities, there will be a problem that the proposed route is not ideal and short.

The results indicated that increasing the number of iterations to 100,000 for different group sizes of cities (10, 50, 100, and 200) leads to longer computation times to find the optimal solution. However, the increased iterations result in more accurate solutions, especially for larger cities. For small cities (10-50 cities), the time taken to reach the optimal solution may range from 60 to 120 s, which is relatively lengthy considering the smaller number of cities. Through these experiments, we can conclude that when the number of cities is large (100, 200), the number of iterations must also be large to obtain more accurate results without visiting the city more than once, but we will face the problem of the time taken to reach the optimal solution, which will be long, but when the number of cities is small, we can use a small number of iterations to get the best results in less time. For this reason, we will change the values for both population size and the number of iterations in the following experiment

Table 2. Comparision of algorithms based on time and distance using a modified genetic algorithm

Cities	Iteration	Population	Distance	Time to complete find the best path
10	1000	50	14.5777	03.10(s)
10	100	100	16.4647	01.20(s)
50	1000	50	136.3001	03.60(s)
50	100	100	144.9281	02.30(s)
100	1000	50	5555.657	03.55
100	100	100	562.394	02.74(s)
200	1000	100	1300.978	17.15(s)
200	5000	100	1650.121	42.31(s)

Table 2 illustrates the experimental results for different combinations of population sizes and iterations for different group sizes of cities. The "Ideal Solution Achieved" column indicates whether the optimal solution was found within a reasonable time frame. Adjustments to population size and iterations have helped in achieving optimal solutions for larger cities within acceptable time limits. By choosing population size (100) and iterations number is (50,000). When choosing the number of repetitions (10,000) and the size of the community (100), no optimum path was obtained.

CONCLUSIONS AND FUTURE WORKS

CONCLUSIONS

Genetic algorithms are used to identify the optimal answer for the traveling salesman problem. In this study, the efficacy of the genetic algorithm heavily relies on the employed operators and the encoding scheme utilized for the problem. Specifically, three mutation strategies—reverse, swapping, and sliding—were investigated to improve algorithm's execution. In optimizing quantity of repeatable variables and the generation size, a significant enhancement in performance was observed. This improvement led to both increased accuracy of the solutions and faster identification of the optimal solution. Overall, the recommended technique demonstrated improved effectiveness in addressing the traveling salesman problem, highlighting the substantial impact of mutation strategies on the algorithm's performance. The previous chapters' testing assessments have been described. In solving TSP, the evolutionary algorithm's strengths and flaws are highlighted, as is its capacity to create pretty excellent solutions in a reasonable amount of time. The study investigated the influence of many variables on GA performance, with the outcome of combining the three cross-over techniques—flipping, swapping, and sliding—being the most significant. In cases where the results were favourable, more exact, accurate, and significantly impacted the genetic algorithm's performance. It was advised that making more modifications to the program or hybridizing it with other algorithms maight boost its performance.

FUTURE WORKS

Improved GA will be utilized to determine the optimal route in the electrical dashboard. In addition, the GA method may be used with other algorithms like the firefly algorithm for solve the TSP issue. He also advised making more modifications to the program.

References

[1] A.-H. Zhou et al., “Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming,” Information,2018, vol. 10, no. 1, p. 7.

[2] A. Kumar, “Improved Genetic Algorithm to Solve Small Scale Travelling Salesman Problem,” Proc. 2020, INDIA, Int. Conf. Intell. Comput. Control Syst. ICICCS 2020, no. Iciccs, pp. 516–520, doi: 10.1109/ICICCS48265.2020.9120880.

[3] J. Zheng, J. Zhong, M. Chen, and K. He, “A reinforced hybrid genetic algorithm for the traveling salesman problem,” Comput. Oper. Res., 2023, vol. 157, p. 106249.

[4] S. Akter, N. Nahar, M. Shahadathossain, and K. Andersson, “A New Crossover Technique to Improve Genetic Algorithm and Its Application to TSP,”2019, USA, 2nd Int. Conf. Electr. Comput. Commun. Eng. ECCE 2019, pp. 7–9, doi: 10.1109/ECACE.2019.8679367.

[5] J. Song, G. Xue, and Y. Kang, “A novel method for optimum global positioning system satellite selection based on a modified genetic algorithm,” PLoS One,2016, vol. 11, no. 3, pp. 1–14.

[6] B. Lin, X. Sun, and S. Salous, “Solving Travelling Salesman Problem with an Improved Hybrid Genetic Algorithm,” J. Comput. Commun.,2016, vol. 04, no. 15, pp. 98–106, doi: 10.4236/jcc.2016.415009.

[7] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, “A hybrid genetic algorithm for the traveling salesman problem with drone,” J. Heuristics,2020, vol. 26, no. 2, pp. 219–247, doi: 10.1007/s10732-019-09431-y.

[8] H. Benbya, T. H. Davenport, and S. Pachidi, “Artificial intelligence in organizations: Current state and future opportunities,” MIS Q. Exec.,2020, vol. 19, no. 4..

[9] A. Pannu and M. T. Student, “Artificial Intelligence and its Application in Different Areas,” Certif. Int. J. Eng. Innov. Technol.,2008, vol. 9001, no. 10, pp. 2277–3754.

[10] S. D. Erokhin, “A Review of Scientific Research on Artificial Intelligence,” 2019 , USA, Syst. Signals Gener. Process. F. Board Commun. SOSG 2019, pp. 1–4, doi: 10.1109/SOSG.2019.8706723.

[11] P. M. Hariyadi, P. T. Nguyen, I. Iswanto, and D. Sudrajat, “Traveling salesman problem solution using genetic algorithm,” J. Crit. Rev.,2020, vol. 7, no. 1, pp. 56–61.

[12] M. A. Al-Furhud and Z. H. Ahmed, “Experimental Study of a Hybrid Genetic Algorithm for the Multiple Travelling Salesman Problem,” Math. Probl. Eng.,2020, vol. 2020, doi: 10.1155/2020/3431420.

[13] A. Uthayasuriyan, H. Chandran, U. V Kavvin, S. H. Mahitha, and G. Jeyakumar, “A Comparative Study on Genetic algorithm and Reinforcement Learning to Solve the Traveling Salesman Problem,” Res. Reports Comput. Sci.,2023, pp. 1–12..

[14] İ. İlhan, “An Application on Mobile Devices with Android and IOS Operating Systems Using Google Maps APIs for the Traveling Salesman Problem,” Appl. Artif. Intell.,2017, vol. 31, no. 4, pp. 332–345, doi: 10.1080/08839514.2017.1339983.

[15] A. Arram and M. Ayob, “A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems,” Comput. Ind. Eng.,2019, vol. 133, pp. 267–274, doi: 10.1016/j.cie.2019.05.012.

[16] J. C. Zhang, “Comparison of various algorithms based on TSP solving,” J. Phys. Conf. Ser.,2021, vol. 2083, no. 3, doi: 10.1088/1742-6596/2083/3/032007.

[17] M.Sami, " Comparative Analysis of the MDORA and AODV Routing Protocols for Vehicles Operating at Constant and Variable Speeds" ,NTU Journal of Pure Sciences, 2024 vol.3,no.3, doi: 10.56286/r39bst03 .