Optimizing Genetic Algorithm for Travelling Salesman Problem by Modifying Parameter

Chan Lik Yin
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP064809@mail.apu.edu.my

Loo Hui Ying
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP065072@mail.apu.edu.my

Dr. Adeline Sneha J
Senior Lecturer/ School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
adeline.john@apu.edu.my

Dr. Kamalanathan Shanmugam
Senior Lecturer/ School of Technology
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
kamalanathan@apu.edu.my

Juhairi Aris Muhamad Shuhili
School of Engineering
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
juhairi.shuhili@apu.edu.my

***Abstract*—The Travelling Salesman Problem (TSP) is a classical optimization algorithm problem in the computer science field. Genetic Algorithm (GA) is an effective technique for solving the TSP. The objective of this research is to find the best possible parameter combination to improve the performance of GA by modifying the parameter values. This study focuses on exploring various combinations of population size, selection method, crossover rate, and mutation rate. The research involves conducting experiments with different parameter combinations to identify the optimal solution for the problem at hand. The findings indicate that a specific set of parameters significantly enhances the performance of the GA, particularly in terms of solution quality and convergence time. To prove that the parameters obtained are effective, standard problems are used to test GA with the tuned and untuned parameters and compare the results. The result shows that GA using the tuned parameters outperformed the untuned parameter. Further research on the exploration of other parameters such as mutation and crossover operators is recommended.**

Keywords—genetic algorithm, travelling salesman problem, parameter modification

I. INTRODUCTION

The Travelling Salesman Problem (TSP) is a popular algorithmic problem in the computer science field. Its goal is to find the shortest path of a specified quantity of cities for a salesperson to travel to all the cities once and return to the departure point (Han, et al., 2023). Finding the most optimal solution has been a challenge for mathematicians and computer scientists for decades. It is not merely an academic problem, but also important in real-life applications such as manufacturing, delivery business, and logistics. A more optimal or ideal route will benefit the delivery or logistic business as the time and fuel consumption can be minimized and thus improve the profitability of businesses. Besides, less travelling distance will also indirectly bring positive effects on the environment and society due to reduced greenhouse gas emissions.

In the field of computer science, the TSP has drawn a lot of attention as it is a common problem used in academics and easy to explain yet challenging to solve. It is known as a nondeterministic polynomial (NP) hard problem, meaning that as the quantity of cities increases, the potential solution sequences grow exponentially. Thus, the solution for the TSP depends on approximation algorithms that attempt various arrangements of cities and then select the shortest path. There are a few well-known algorithms that can be used to solve this problem such as the branch and bound algorithm and the Ant Colony Optimization. However, in this research, we use the Genetic Algorithm (GA) as it is also an effective strategy for TSP.

Genetic Algorithm (GA) is known as an optimization technique for improving potential solutions in order to solve complicated problems such as the TSP in the Artificial Intelligence (AI) domain. It was first introduced in the 1960s by John Holland, which is inspired by Darwin's theory of natural evolution. It was first developed to simulate the natural selection and genetics processes, this has laid the foundation for this algorithm to solve optimization problems that are challenging using traditional computational methods.

GA executes by continuously evolving the potential solutions in a population to obtain the optimal or nearly optimal solution for a problem, such as finding the shortest path for the TSP. It mimics the natural selection process in which the fittest individuals are chosen to reproduce the next generation of offspring that inherit their parents' characteristics, and they will have better fitness compared to their parents. At the end of this iteration, the fittest individuals will be obtained. (Yang, et al., 2023)

There are a few parameters of GA, which are the crossover rate, mutation rate, population size, and generation number. The parameters play a significant role in shaping the performance of the algorithm. An appropriate set of parameters can help to produce the best result with a shorter convergence time. Thus, the objective of this study is to optimize the performance of the GA for solving the TSP by modifying parameters. The results for each set of parameters were recorded to be compared and find out the parameter combination that produced the best result.

II. LITERATURE REVIEW

A. Similar Projects

Several studies related to our topic have been conducted by past researchers such as using Genetic Algorithm (GA) to solve the Travelling Salesman Problem (TSP). The methodologies used by past researchers give us an insight into the approaches to modify parameters. By summarizing the findings from these studies, we can have a better understanding of how to optimize the performance of the algorithm.

Yue and Wang (2015) have proposed an improved Ant Colony Optimization (ACO), CEULACO algorithm to overcome the limitation of classical ACO in solving the TSP. The CEULACO algorithm has improved the pheromone concentration, pheromone evaporation rate, pheromone updating rule, and search strategy from the classical ACO algorithm. The performance of CEULACO has proved to be the best among the comparison with ACO and IMACO in solving 10 TSP instances.

Gunduz and Aslan (2021) have developed a discrete Jaya algorithm (DJAYA), an improved Jaya algorithm that has modified its parameters and operators to solve discrete optimization problems including TSP. The authors generate initial solutions using random permutations and the nearest neighbourhood approach. The transformation operators in the Jaya algorithm have been replaced by a combination of transformation operators which are swap, shift, and symmetry according to roulette wheel selection. The search tendency parameter has been modified after an analysis of the parameter is tested on a benchmark instance.

Hatamlou (2018) has applied the black hole algorithm (BH), an algorithm that simulates the black hole to solve optimization problems such as TSP. The BH algorithm is based on the black hole in space which has an extremely high gravitational power. The BH algorithm will attract the population of solutions towards the best solution to form a better solution. The result after a comparison with several algorithms shows efficiency and robustness in solving huge TSP with high accuracy and low standard deviation.

B. Comparison with Other Algorithms

Meta-heuristics algorithms are high-level algorithms that optimize a problem by improving a number of solutions. Meta-heuristics algorithms rarely make assumptions about the problem, resulting in their reliability in solving large spaces of problems (Desale, et al., 2015). As one of the popular NP-hard problems in combinatorial optimization, TSP has been solved by many meta-heuristic algorithms including GA. Each algorithm possesses its unique strengths such as fast execution time, accurate solution, and low variance. Numerous algorithms have been compared to find out the strengths of respective algorithms and determine if is GA the suitable algorithm for this study.

Rao T. S. (2017) has made a comparison between GA, Simulated Annealing Algorithm (SA), and Nearest Neighbour Algorithm (NN) to find the shortest route generated in solving 5 instances of TSP. GA shows its high accuracy in small numbers of cities by outperforming the other algorithms by 10% on average.

Another study conducted by Halim and Ismail (2017) compared the performance of GA, SA, and NN. Tabu Search (TS), Ant Colony Optimization (ACO), and Tree Physiology Optimization (TPO) in solving 15 benchmark TSPs. The result from this study shows that GA has one of the fastest computational times when finding optimal routes for all sizes of nodes. GA also consistently generates solutions that are close to the optimum route.

Chaudhari and Thakkar (2019) have applied GA, ACO, Particle Swarm Optimization (PSO) Algorithm, Artificial Bee Colony (ABC), and Firefly Algorithm (FA) to solve three benchmark TSP. The result shows that GA is one of the most consistent in providing near-optimal routes for all the TSPs.

C. Optimization Techniques

Several techniques can be used to optimize and improve GA for solving the TSP and the techniques have been proven to be effective by past researchers. The techniques include tuning parameters, using greedy approaches, and hybridizing with Particle Swarm Optimization (PSO).

The efficiency of the algorithm depends on the algorithm coding, operators, and settings of the parameters. Therefore, there are some research found to be conducted to examine the effect or impact of modifying the GA parameters and operators including mutation rate and initial population rate for the TSP to reduce the convergence time and find the best result. The results from both studies showed that tuning parameters and operators can improve the performance of GA for solving TSP. (Rexhepi, et al., 2013; Mosayebi, M., & Sodhi, M., 2020)

Rana and Srivastava (2017) have improved GA to solve the TSP by integrating a greedy approach to the original GA in terms of generating chromosomes and proposing a new greedy crossover operator. It can help to search the solution space deeper and explore solutions with better fitness. The findings demonstrated that the GA incorporating greedy approaches exhibits superior performance in path length compared to other algorithms.

The other technique was proposed by Gupta, et al. (2018) which is hybridizing with the PSO algorithm. The objective of this study is to exploit the higher convergence rate of PSO to GA. Ten standard problems were used to test the proposed algorithm. Results showed that the hybrid GA-PSO algorithm performed better than the original GA and PSO.

D. Methodology/Approach

In this section, an overview of the methodologies used in the studies and research mentioned above is summarized and provided. These methods that have been applied in similar projects will be useful as a guide for us in our research.

Rexhepi, et al. (2013) analyzed different solutions from the outcome using distinct initial populations and mutation rates. They set the maximum generation number to 10,000 with initial population sizes of 1000, 5000, and 10000. Every population size was tested with 1%, 3%, 5%, and 10% mutation probability. Then, the results for the same population size with multiple mutation rates and results for the same mutation rate with different initial populations were plotted on a graph.

Mosayebi and Sodhi (2020) conducted the study by using the Design of Experiment (DOE) methodology. They selected three popular crossover operators which were one-point, two-point, and Cycle, and two mutation operators which were

Inversion and Swap. The population size was generated with a relative population rate of 10 and 20. There were two levels of parameter settings. They carried out pilot runs with the gr17 problem to identify the relationship between the fitness value and other parameters by establishing a regression equation. However, the results for one-point and two-point operators were close to each other. Therefore, a second experiment was conducted based on the result from the first experiment. Then, the authors tested the tuned parameters on standard problems. The results showed that using the tuned parameters to solve the TSP has a better result.

E. Conclusion/Recommendation

In conclusion, GA can be used to solve the TSP, and has been proven by past researchers. It can produce an optimal solution for the problem.

It is proven that GA is an effective and efficient algorithm for solving the TSP when compared to other algorithms in terms of accuracy, consistency, and computational speed. Thus, GA will be used to solve the TSP in this study.

Several optimization techniques were proposed by past researchers and were proven to be effective for GA to solve the TSP. One of them was by tuning parameters such as the mutation rate and initial population size. This literature survey also reviewed the methodologies used by other researchers to modify parameters. Therefore, in this paper, GA parameters will be tuned to find the best potential value to solve the TSP.

III. MATERIALS AND METHODS

A. Algorithm Implementation

Genetic Algorithm (GA) is a search-based optimization algorithm that is commonly applied to obtain optimal or nearly optimal solutions for difficult problems like the Travelling Salesman Problem (TSP). In GA, there are some possible solutions in a population for a given problem initially. Then, these solutions will undergo crossover and mutation processes to produce new offspring, and this process iterates over several generations until it meets the termination condition. Every solution has a fitness value, and a fitter solution has a greater chance to mate and yield more fitter solutions. Thus, the solutions will keep improving over generations.

1) Source Code

The GA source code that we found to solve the TSP is in the Python programming language. It is provided by hassanzadehmahdi (2021) on the GitHub website. To run the code, simply download from GitHub and extract the zip file. Run the tsp.py in any text editor or IDE's terminal and the result will be displayed including the number of generations in runs and the best path length, Besides, a window showing the solution will pop up.

The code starts with getting the position of cities from the text file "TSP51.txt". The initial population will be generated based on the population size. The population is generated randomly and sorted in ascending order based on its distance.

After the initial population is generated, the population will evolve for 200 generations, or the target value is achieved. The crossover rate will determine the probability of crossover for each population. If the crossover occurs, two-parent chromosomes will be selected through tournament selection with 4 tournament sizes. One-point crossover operator will randomly select a crossover point on both parent chromosomes, and the information to the right of the point will be swapped to reproduce the child chromosomes. If the crossover does not occur, the child's chromosomes will be a copy of random parent chromosomes.

After the child's chromosome is reproduced, the mutation will randomly perform based on the mutation rate. If the mutation occurs, the swap mutation operator will select two random points for both child chromosomes. The two points will be swapped and the child chromosome will be mutated. All the child chromosomes will be the new population and sorted in ascending order. Based on elitism, the top two child chromosomes will be kept for the next generations.

After 200 generations, the best solution will be plotted in a 2D graph with the cities and the path will be visualized.

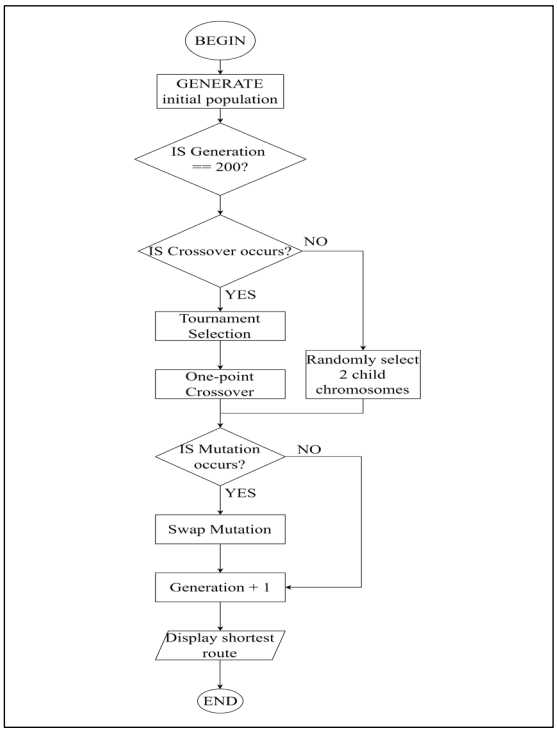

Fig. 1. Flow chart of Genetic Algorithm

After the optimization of the parameters, the best combination of parameters that has the best performance on average will be applied to two standard TSPs to test its performance. The standard TSPs are dantzig42 and eil101, from the TSPLIB that are contributed by Gerhard Reinelt (Reinelt, 1991). The result will be compared with the optimal path length provided by the author to test the efficiency of the tuned parameter.

2) Purpose

The main purpose of implementing this algorithm is to solve various sizes of the TSP instance efficiently. The original parameters set by the source code owner are tuned to find a better solution for the problem which is the shortest possible path. It aims to contribute to domains such as delivery and logistics businesses to shorten their travelling distance.

3) Parameters

The values of GA parameters including population size, selection of operators, and operator rates are still an open question to be used in an arbitrary problem. They are explored to reduce the time required to get the best optimal solution. The basics of the code do not change throughout the study, but the parameters will be modified to improve the performance of the algorithm (Rexhepi, et al., 2013).

In the source code by hassanzadehmahdi, the default parameter settings are as follows: swap mutation operator, 1-point crossover operator, tournament selection method, tournament selection size of 4, 0.1 mutation rate, 0.9 crossover rate, 2000 for population size and 200 for maximum generations.

The mutation and crossover operator, tournament selection size, and maximum generations will be maintained throughout the study, while the parameters that will be tuned concurrently to optimize the algorithm for the TSP are as below:

a) Mutation Rate

Mutation is a GA technique used to generate new genetic material. It can help to maintain the diversity of genetics from one generation to the other. The mutation rate is the probability of an individual in the population undergoing mutation. A mutation rate with a greater value helps to explore deeper to the search space but might cause the algorithm to get stuck in a suboptimal solution. (Datta, 2023)

b) Crossover Rate

Crossover is a process that produces a new solution by combining the best features of two parents which also helps in producing genetic diversity. The crossover rate is the likelihood of the occurrence of the crossover process. The optimal value for this parameter depends on the problem as well as the population. A high crossover rate ensures the exploration of search space is sufficient but will lead to a longer search time. (Datta, 2023)

c) Population Size

GA starts with an initial population which consists of potential solutions to the TSP. Each solution is an individual, and the number of individuals in the population is set with the population size parameter and they are generated randomly in this source code (Kanade, 2023).

d) Selection Method

The selection method is the most crucial parameter that will affect the performance of GA. Its function is to select a pair of parents to generate a new solution for the next generation. A selection method aims to exploit the best features of the parent solutions with good features to improve the solutions for the next generation. Some popular selection methods are Roulette Wheel Selection, Stochastic Universal Sampling, and tournament selection. Tournament selection is the default selection method used in the source code and in this study. (Jebari & Madiafi, 2013)

B. Hardware Requirement

There is no high hardware requirement for a computer to run the code. A computer with a quad-core processor such as Intel Code i5 with at least 4GB of RAM is suggested. Besides, a few gigabytes of free storage space will be sufficient to download the source code and datasets for other standard problems to test the code if needed. In addition, a basic graphics card is also enough for running the code as it mainly depends on the CPU. Overall, the requirement for hardware to run this program is relatively lightweight and can be run on most modern computers.

C. Software Requirement

There are a few software requirements to run the code. The most important is to download the source code posted by hassanzadehmahdi from GitHub. The folder provided includes the main Python script and a dataset. Besides, the computer must be installed with Python with 3.x versions as the code is written in Python. It is available on the official Python website. After that, the necessary Python library that is used in the code should be installed via Command Prompt for Windows users and Terminal for macOS or Linux users, with the code 'pip install matplotlib'. Apart from that, a text editor such as VS code and Sublime Text, or an Integrated Development Environment (IDE) like PyCharm or IDLE is required to run the script. These software requirements with the Python environment set up correctly will be essential to run the code.

IV. RESULTS AND DISCUSSION

A. Discussion on Implementation

The parameters that are modified in this study included population size, selection method, crossover rate, and mutation rate. The other parameters remain as default which are the swap mutation operator, one-point crossover operator, 200 for maximum generation number, and 4 for the tournament selection size if the tournament selection method is used. The target length of the optimal path is 450, which was set by the source code author. If the code achieves the target length before reaching the maximum generations, the code will terminate.

The code was run with different combinations of parameters to get the best potential solution for the TSP. The parameter modifications are as follows: The population sizes were 1000, 2000, and 3000. For each of the population sizes, it was run with a crossover rate (0.1, 0.5, 0.7, 0.9) and a mutation rate (0.1, 0.3, 0.7, 0.5, 0.9) for each combination of population size and crossover rate. Then, the tournament selection method and Rank Selection were used as the selection method for every set.

Each parameter set was run for ten times. The average path length, standard deviation, and average computational time were calculated and recorded in tables.

After getting the best parameter combination for the TSP, it was tested with a standard problem dantzig42 and eil101 to be compared with the result produced by the algorithm with untuned parameters.

B. Results

The results were recorded in tables according to the population sizes. CR stands for crossover rate, Avg stands for average which is the average path length, and time means the average computational time for the parameter set.

TABLE I. RESULTS OF 1000 POPULATION SIZE

Selection Method	CR	Mutation Rate				
		0.1	*0.3*	*0.5*	*0.7*	*0.9*
Tournament	0.1	Avg:970.27 Time:5.63	Avg:1128.42 Time:5.72	Avg:1171.01 Time:6.30	Avg:1186.42 Time:6.21	Avg:1212.76 Time:5.82
	0.5	Avg:546.18 Time:7.51	Avg:545.56 Time:7.56	Avg:716.73 Time:8.24	Avg:843.19 Time:7.87	Avg:909.82 Time:7.99
	0.7	Avg:522.37 Time:8.86	Avg:511.22 Time:8.89	Avg:538.09 Time:8.83	Avg:699.09 Time:9.27	Avg:798.00 Time:9.41
	0.9	Avg:529.48 Time:9.19	Avg:508.90 Time:9.33	Avg:519.34 Time:9.62	Avg:547.24 Time:9.79	Avg:704.90 Time:9.24
Rank Selection	0.1	Avg:661.50 Time:10.83	Avg:745.16 Time:10.70	Avg:854.60 Time:9.63	Avg:952.25 Time:10.11	Avg:1021.63 Time:11.14
	0.5	Avg:543.39 Time:21.61	Avg:545.13 Time:20.05	Avg:537.96 Time:21.61	Avg:551.88 Time:23.93	Avg:570.33 Time:21.27
	0.7	Avg:523.43 Time:20.68	Avg:516.75 Time:22.35	Avg:512.17 Time:23.33	Avg:543.33 Time:23.83	Avg:536.30 Time:24.86
	0.9	Avg:519.85 Time:31.23	Avg:526.95 Time:33.10	Avg:521.81 Time:32.37	Avg:**503.73** Time:34.22	Avg:524.45 Time:54.31

TABLE II. RESULTS OF 2000 POPULATION SIZE

Selection Method	CR	Mutation Rate				
		0.1	*0.3*	*0.5*	*0.7*	*0.9*
Tournament	0.1	Avg:955.221 Time:13.74	Avg:1074.63 Time:13.88	Avg:1143.90 Time:13.01	Avg:1167.59 Time:17.41	Avg:1174.22 Time:14.41
	0.5	Avg:502.24 Time:18.97	Avg:528.97 Time:17.16	Avg:693.39 Time:17.48	Avg:794.68 Time:17.67	Avg:892.66 Time:21.76
	0.7	Avg:513.76 Time:18.31	Avg:481.74 Time:18.21	Avg:506.87 Time:18.63	Avg:645.46 Time:18.13	Avg:767.73 Time:18.50
	0.9	Avg:488.28 Time:22.90	Avg: 483.67 Time:26.44	Avg:**478.00** Time:22.38	Avg:535.57 Time:23.04	Avg:677.66 Time:22.73
Rank Selection	0.1	Avg:595.79 Time:20.81	Avg:650.20 Time:22.70	Avg:723.53 Time:22.39	Avg:839.35 Time:22.45	Avg:931.35 Time:24.43
	0.5	Avg:512.81 Time:55.02	Avg:499.03 Time:61.23	Avg:508.27 Time:64.13	Avg:533.77 Time:66.35	Avg:540.08 Time:67.23
	0.7	Avg:513.66 Time:69.02	Avg:504.76 Time:78.73	Avg:494.19 Time:86.03	Avg:512.77 Time:91.69	Avg:508.43 Time:95.10
	0.9	Avg:524.27 Time:77.70	Avg:508.34 Time:90.70	Avg:504.16 Time:100.91	Avg:510.61 Time:107.44	Avg:502.65 Time:111.06

TABLE III. RESULTS OF 3000 POPULATION SIZE

Selection Method	CR	Mutation Rate				
		0.1	*0.3*	*0.5*	*0.7*	*0.9*
Tournament	0.1	Avg:902.31 Time:16.67	Avg:1085 Time:17.95	Avg:1123.11 Time:16.83	Avg:1167.31 Time:18.24	Avg:1189.79 Time:18.10
	0.5	Avg:496.95 Time:24.48	Avg:482.75 Time:25.63	Avg:652.24 Time:24.11	Avg:779.21 Time:27.09	Avg:851.13 Time:25.28
	0.7	Avg:506.40 Time:28.83	Avg:474.86 Time:27.91	Avg:508.65 Time:28.39	Avg:643.29 Time:29.02	Avg:751.63 Time:27.85
	0.9	Avg:501.61 Time:33.81	Avg:477.43 Time:34.45	Avg:**469.40** Time:35.54	Avg:539.76 Time:38.03	Avg:647.39 Time:38.78
Rank Selection	0.1	Avg:557.73 Time:39.90	Avg:601.84 Time:40.77	Avg:705.03 Time:41.34	Avg:770.55 Time:41.52	Avg:882.85 Time:41.62
	0.5	Avg:513.10 Time:113.75	Avg:510.04 Time:130.48	Avg:505.75 Time:138.07	Avg:521.86 Time:141.94	Avg:515.03 Time:144.43
	0.7	Avg:513.77 Time:140.39	Avg:497.18 Time:192.82	Avg:489.11 Time:183.25	Avg:488.38 Time:202.66	Avg:510.46 Time:222.43
	0.9	Avg:511.39 Time:165.03	Avg:486.20 Time:193.31	Avg:487.83 Time:217.76	Avg:502.02 Time:234.71	Avg:500.82 Time:275.54

Based on Table I, the best parameter for 1000 population size for solving the TSP was using the Rank selection method, 0.9 crossover rate, and 0.7 mutation rate. For the 2000 population size, the parameter settings for the best potential solution were the Tournament selection method, 0.9 crossover rate, and 0.5 mutation rate while the Tournament selection method, 0.5 crossover rate, and 0.9 mutation rate are found to be the most optimal for 3000 population size. Overall, the best potential solution was using a 3000 population size, Tournament selection method, 0.9 crossover rate, and 0.5 mutation rate.

Based on the result of the comparison, it is significantly showing the population, crossover rate, mutation rate, and selection method are affecting the result and the execution time.

As the population increases, the average shortest path length will decrease greatly while the average execution time will increase gradually. If the population is large, the probability of finding better solutions will increase as the exploration of the path has been enhanced. However, the large exploration will lead to a high cost of computational time.

As the crossover rate increases, the overall result will be shorter, but the execution time will be longer. A high crossover will increase the exploration rate and cause the population more homogeneous as the population is sharing genetic information frequently. Nevertheless, a high crossover rate will increase the number of crossover operations, contributing to high computational costs.

The impact of the mutation rate in the GA is a complex interplay with the crossover rate. When the mutation rate increases, the average execution time slightly increases as the probability of mutation operation has increased. However, when the crossover rate is high, the 0.5 mutation rate shows a shorter path length than the other mutation rate. The moderate mutation rate shows the balance between exploration and exploitation by allowing sufficient exploration and preserving promising solutions. The high crossover rate provides wide exploration for the high mutation rate to increase diversity. When the mutation is excessively high, the algorithm has too much randomness which leads to the disruption of good solutions.

While the rank selection method may not yield the best results in the comparison, it did exhibit distinct characteristics when compared to the tournament operator. The rank selection method was performing stably with less sensitivity to the parameters. However, it took much longer than the tournament operator when the mutation and crossover rate was high. The rank selection method is a complex calculation to select the parent chromosome. Therefore, high mutation and crossover rates will constantly generate a diverse population that increases the rate of computing the selection method.

Some standard problems, dantzig42 and eil101, were used to test the effectiveness of the tuned parameters by comparing them with the result of the untuned parameters. The tuned and untuned parameters will be run ten times to calculate the average path length and computational time. The graph below shows the differences in the path of running using tuned and untuned parameters.

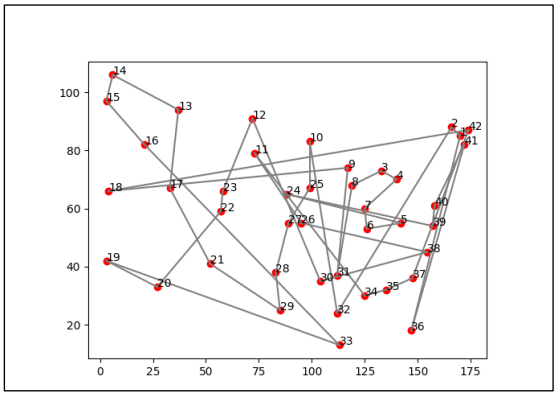

Fig. 2. Results of untuned parameters in solving dantzig42

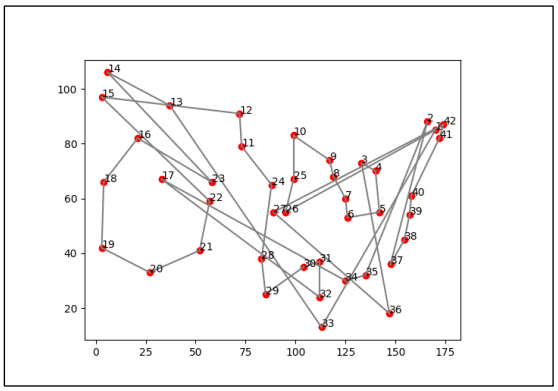

Fig. 3. Results of tuned parameters in solving dantzig42

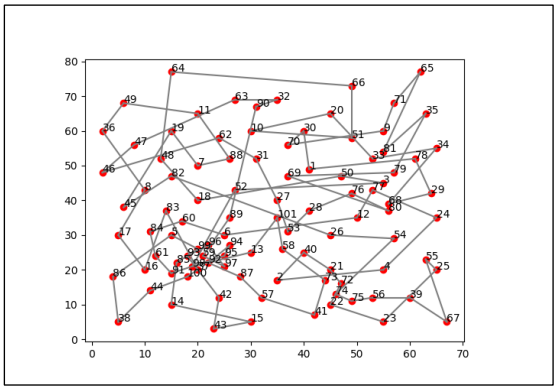

Fig. 4. Results of untuned parameters in solving eil101

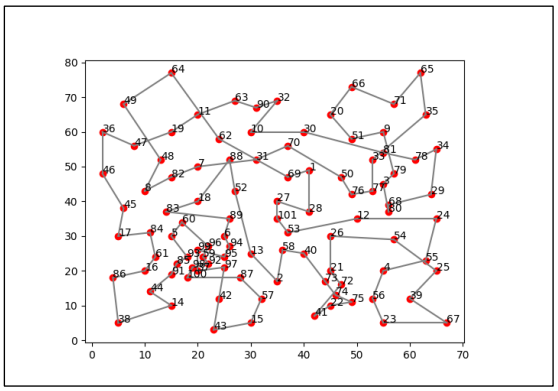

Fig. 5. Results of tuned parameters in solving eil101

TABLE IV. RESULTS OF TUNED AND UNTUNED PARAMETERS

Standard Problem	Parameter	Average path length	Average computational time	Difference from optimal path
Dantzig42	Untuned	777.19	21.15	11.19%
	Tuned	721.01	26.69	3.15%
Eil101	Untuned	1052.70	63.75	67.36%
	Tuned	997.16	91.75	58.51%

Based on Table IV, the tuned parameters, with a 3000 population size, Tournament selection method, 0.9 crossover rate, and 0.5 mutation rate, highly outperformed the untuned parameters, which have a 2000 population size, Tournament selection method, 0.9 crossover rate, and 0.1 mutation rate. In the dantzig42 problem, the performance of tuned parameters has exceeded the untuned parameters by 7.24%. In the eil101 problem, the result of the tuned parameters is 5.28% shorter than the untuned parameters. Although the tuned parameters required a longer computational time to generate the path, it is still within an acceptable range of computational time. The result for the standard problem eil101 has shown that the algorithm is not suitable for a large number of cities. However, the GA is considered one of the best algorithms for solving the TSP with a small number of cities.

V. CONCLUSION

In conclusion, the Genetic Algorithm (GA) is an effective technique to solve the Travelling Salesman Problem (TSP). The best parameter combination that has been obtained is 3000 population size, 0.5 mutation rate, 0.9 crossover rate, tournament selection method, with swap mutation operator and one-point crossover operator. It has proven to be effective when compared to the untuned parameters on the standard problems. It does improve performance in terms of path length. Modifying parameters to find the best combination is important to improve the performance of the algorithm for diverse problem instances including for the delivery and logistics business. Further research can be conducted by exploring other parameters such as changing the crossover and mutation operator to examine the impacts on the performance of GA for the TSP.

ACKNOWLEDGMENT

We would like to thank all authors and other School of Computing members who contributed to this study. Besides, we would like to express our gratitude to the source code and datasets owner for their source code and datasets.

REFERENCES

Datta, S. (2023, June 17). *Genetic Algorithms: Crossover Probability and Mutation Probability*. Retrieved from https://www.baeldung.com/cs/genetic-algorithms-crossover-probability-and-mutation-probability#:~:text=Mutation%20probability%20is%20a%20parameter,is%20mutated%20at%20each%20generation.

Desale, et al. (2015). Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World:

A Survey. *International Journal Of Computer Engineering In Research Trends, Volume 2, Issue 5*, 296-304.

Gunduz, M., & Aslan, M. (2021). DJAYA: A discrete Jaya algorithm for solving traveling salesman problem. *Applied Soft Computing*, 1-15.

Gupta, I. K., Shakil, S., & Shakil, S. (2018). A Hybrid GA-PSO Algorithm to Solve Traveling Salesman Problem. *Advances in Intelligent Systems and Computing*, 453-462.

Halim, A. H., & Ismail, I. (2017). Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem. *Arch Computat Methods*, 367-380.

Han, T. R., Zer, B. Y., Lun, N. C., Yi, G. F., Jie, W. J., & Salam, Z. A. (2023). Optimizing ACO Algorithm for the TSP. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304)*, 52-56.

hassanzadehmahdi. (2021). *Traveling-Salesman-Problem-using-Genetic-Algorithm*. Retrieved from GitHub: https://github.com/hassanzadehmahdi/Traveling-Salesman-Problem-using-Genetic-Algorithm

Hatamlou, A. (2018). Solving travelling salesman problem using black hole algorithm. *Methodologies and Application*, 8167-8175.

Jebari, K., & Madiafi, M. (2013). Selection Methods for Genetic Algorithms. *Int. J. Emerg. Sci.*, 333-344.

Kanade, V. (2023, September 6). *What Are Genetic Algorithms? Working, Applications, and Examples*. Retrieved from https://www.spiceworks.com/tech/artificial-intelligence/articles/what-are-genetic-algorithms/

Kinjal, C., & Ankit, T. (2019). Travelling Salesman Problem: An Empirical Comparison Between ACO, PSO, ABC, FA and GA. *Emerging Research in Computing, Information, Communication and Applications. Advances in Intelligent Systems and Computing, vol 906.*

Mosayebi, M., & Sodhi, M. (2020). Tuning Genetic Algorithm Parameters using Design of Experiments. *GECCO*, 1937-1944.

Rana, S., & Srivastava, S. R. (2017). Solving Travelling Salesman Problem Using Improved Genetic Algorithm. *Indian Journal of Science and Technology*, 1-6.

Rao, T. S. (2017). A Comparative Evaluation of GA and SA TSP in a Supply Chain Network. *Materials Today: Proceedings 4*, 2263-2268.

Reinelt, G. (1991). TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing, 3*, 376-384.

Rexhepi, A., Dika, A., & Maxhuni, A. (2013). Analysis of the Impact of Parameters Values on the Genetic Algorithm for TSP. *IJCSI International Journal of Computer Science Issues*, 158-164.

Yang, C. M., Pek, V., Ling, S. H., wei, T. C., & Salam, Z. A. (2023). Solver of 8-Puzzle with Genetic Algorithm. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304)*, 28-32.

Yue, Y., & Wang, X. (2015). An Improved Ant Colony Optimization Algorithm for Solving TSP. *International Journal of Multimedia and Ubiquitous Engineering*, 153-164.