ENHANCED KNOWLEDGE IN SCIENCES AND TECHNOLOGY
e-ISSN: 2773-6385

EKST

Vol. 4 No. 1 (2024) 171-182
https://publisher.uthm.edu.my/periodicals/index.php/ekst

Optimal Parameter Value of Genetic Algorithms for Different Size Instances in Solving Traveling Salesman Problem

Nor Ain Shafiqah Abdullah[1], Siti Noor Asyikin Mohd Razali[1*]

[1] *Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia, Kampus Cawangan Pagoh. Hab Pendidikan Tinggi Pagoh, KM 1, Jalan Panchor, 84600 Pagoh, Muar, Johor, MALAYSIA*

*Corresponding Author: asyikinr@uthm.edu.my
DOI: https://doi.org/10.30880/ekst.2024.04.01.020

Article Info

Received: 27 December 2023
Accepted: 3 June 2024
Available online: 27 July 2024

Keywords

Parameter Tuning, Genetic Algorithm, Metaheuristics, TSP, Optimization, Elitism, Crossover, Mutation

Abstract

This study focuses on optimizing the parameters of the Genetic Algorithm (GA) to solve the TSP. Besides, this research is helpful to real-world applications because it offers effective, scalable, and flexible solutions to challenging optimization problems. The contribution of approaches applied in the real world will enhance decision-making, save costs, and enhance efficiency in various fields including finance, engineering, logistics, transportation, manufacturing, and healthcare. Moreover, TSP instances were selected in this study based on the differences in dataset sizes and their uniqueness, which had not been used in previous research. Additionally, genetic operators such as Elitism, order crossover (OX), and swap mutation were employed in the GA. The objective of this study is to analyze the impact of parameter tuning on the performance of the GA by varying the values of population size, number of generations, crossover rate, and mutation rate. Several experiments were conducted, and the optimal parameter values such as population size of 100, 750 generations, and crossover and mutation rates (CMR 2 = [0.9, 0.01] and [0.7, 0.01]), were determined in this study. Subsequently, these values were compared with the values of the common approach used in previous research, which is CMR 1 = [0.9, 0.03]. The study results indicate that the best combination is CMR 2 which outperforms CMR 1 in finding the minimum travel distance and achieving a shorter computation time. Across datasets with varying characteristics for each instance, it was found that a high crossover rate (0.9) contributes to better algorithm accuracy compared to a low crossover rate (0.7). However, TSP instances with larger sizes demonstrated the ability to identify optimum travel distances using a lower crossover rate (0.7). Furthermore, an increase in mutation rate does not necessarily contribute to find the best solution. Nevertheless, the CMR 2 combination, with a lower mutation (0.01) rate compared to CMR 1 (0.03), has contributed to better solutions, especially in the TSP instances examined in this research. In future research, the paper could be extended by conducting comparative analyses between GA and other optimization methods, such as Particle Swarm Optimization (PSO), Simulated Annealing (SA), or Ant Colony Optimization (ACO), to investigate the performance of various applied metaheuristics.

1. Introduction

The Travelling Salesman Problem (TSP) is a classic combinatorial optimization problem that has a wide range of real-world applications [1, 2]. TSP as an NP-hard problem requires finding the most efficient route that visits a collection of cities exactly once and returns to the origin while minimizing the total distance traveled and computational time [3]. In the context of the TSP, instances are specific problem scenarios or datasets that define the cities and the distances between them [4]. These instances are used to test and compare the performance of algorithms. As the number of cities increases, it becomes computationally difficult to solve the TSP and traditional algorithms may struggle to discover optimal or near-optimal solutions in a reasonable amount of time [5].

According to [6], many metaheuristics techniques for solving approximate solutions have been born in succession to better solve the TSP of large search space such as genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), tabu search (TS), and simulated annealing (SA). So, GA has been chosen and applied in this study because due to past research, GA has been effectively used to solve numerous optimization problems including complex TSP due to its broad application since it is a population-based technique where the algorithm is inspired by natural selection that uses reproduction, mutation, and selection to find near-optimal solutions to a problem in reasonable time [7, 8]. Since this study involves small to large datasets, Integer Linear Programming (ILP) is not suitable because it will take longer time consumption for large datasets to obtain optimal solutions for the complex problem size [9]. In the setting of TSP, a population of candidate solutions (individual routes) evolves over generations, with the fittest individuals, representing shorter routes, being more likely to pass on their genetic material to the next generation [10].

Additionally, parameter tuning with genetic algorithms (GAs) proves to be an efficient method for enhancing the performance of TSP solvers across a wide range of instances, as many researchers have successfully proposed improvements to GA approaches [11]. Therefore, the effectiveness of GAs in solving TSP is strongly dependent on the appropriate tuning of algorithmic parameters. That is why, it is important to consider the determination of parameter values including population size, crossover rate, mutation, and termination criteria to balance between exploration and exploitation while ensuring that the algorithm efficiently converges to a high-quality solution. Further, many researchers in the previous studies have made various comparisons of metaheuristic approaches such as Genetic Algorithm (GA) and Ant Colony Optimization (ACO), Simulated Annealing (SA) and ACO and Particle Swarm Optimization (PSO), and ACO to find the best solution of the TSP instances [12, 13]. Then, the impact of parameter values becomes more obvious when dealing with TSP instances of varying sizes due to the complexity of the instances while smaller instances may necessitate different parameter settings than larger instances due to changes in the search space and solution difficulty. Thus, adopting a one-size-fits-all approach to parameter tuning may not be optimal across diverse TSP instances. xx Other previous studies were done in using forecasting method with the statistics technique in worldwide according to various fields of studies [14, 15, 16, 17].

Moreover, the objectives of this study are to analyse the effect of varying population sizes and number of generations on the performance of the GA for solving the TSP across various instance sizes. Next, to determine the optimal combination of crossover and mutation rates (CMR) that result in minimized travel distance for four TSP instances at a fixed population size and number of generations. Lastly, to compare the performance of optimal parameters obtained from this study with a common approach by previous studies for different TSP instances. This paper will discover the impact of parameter tuning on the performance of GA to obtain the lowest travel length and computational time across four TSP instances that have not been used in previous research [18]. Besides, this research is helpful to real-world applications because it offers effective, scalable, and flexible solutions to challenging optimization problems. The contribution of approaches applied in the real world will enhance decision-making, save costs, and enhance efficiency in various fields including logistics, finance, engineering, transportation, manufacturing, healthcare, and many other fields where optimizing resource utilization and minimizing costs is crucial.

2. Methodology

2.1 Dataset Selection

In this study, four TSP instance datasets, namely dantzig42, xqf131, xit1083, and djb2036, have been taken from the Traveling Salesman Problem library (TSPLIB) retrieved from https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html and a very large-scale integration dataset (VLSI) retrieved from https://www.math.uwaterloo.ca/tsp/vlsi/ to represent diverse problem complexities. The number associated with the TSP name refers to the number of cities contained in each instance. Table 1 shows the summary of instances used.

Table 1 *Summary of instances used by Genetic Algorithm*

Enhanced Knowledge in Sciences and Technology Vol. 4 No. 1 (2024) p. 171-182

Instance	dantzig42	xqf131	xit1083	djb2036
Source	TSPLIB	VLSI Dataset	VLSI Dataset	VLSI Dataset
Number of datasets	42 cities	131 cities	1083 cities	2036 cities

2.2 Genetic Algorithm Operation Process

In GA, chromosome representation, fitness selection, and biologically inspired operators are essential for locating optimal solutions [19]. Table 2 shows the analogy of GA terminology to nature.

Table 2 *Analogy of AG terminology to nature*

Nature	GA
Chromosome	String
Gene	Character
Locus	String position
Genotype	Population
Phenotype	Decoded structure

2.3 Initial Population

The initial of the GA for general basis outline, a population is created by generating and encoding *n* random sets of solutions into chromosomes using real-coded representations. The population size is determined by the user of the application. To avoid repeating cities, a list of ordered city numbers is randomly generated. For example, there are 8 cities in a certain area [20], and the chromosome representation is shown in Table 3.

Table 3 *Example of chromosome representation*

2	3	1	5	6	4	8	7

Based on Table 3, the path of this chromosome is 2→3→1→5→6→4→8→7 which shows that the salesman traveled from city 2 to city 3 until city 7.

2.4 Fitness Evaluation

TSP, as it is known, seeks to find a route that travels between each city exactly once and eventually returns to the starting city. The distance between cities is a crucial factor in determining the fitness or quality of a potential solution in the GA. The formula is shown in eq. (1).

$$D_i = \sqrt{x^2 + y^2} \tag{1}$$

where:

i = 1, 2,3, . . . n
D_i = Distance between cities
x = x – coordinate of the cities in the map
y = y – coordinate of the cities in the map

Next, fitness evaluation aims to minimize distance. So, the fitness function will be used as a solution evaluation in TSP by calculating the total distance of the visit represented by the chromosome. The formula is shown in eq. (2).

$$Fitness = \frac{1}{\sum_{i=1}^{n} D_i} \tag{2}$$

where:

n = Number of population size
D_i = Distance between cities

As a result, chromosomes with shorter distances for visits will have higher fitness values, thereby increasing their chances of being selected as parents.

2.5 Selection Operator

During the selection stage, the selection operator of GA identifies individuals with desirable traits, which leads to better solutions. The purpose of selection is to prioritize individuals with higher fitness values with the expectation that their offspring will inherit beneficial traits and improved fitness values. Elitism is a technique that involves initially replicating a select number of the highest-scoring chromosomes into the new population before proceeding to generate the remaining population. Through elitist selection according to [21], the GA preserves the best solutions, preventing the loss of valuable genetic material. Figure 1 below shows the Elitism steps after fitness values are obtained in the fitness evaluation section.

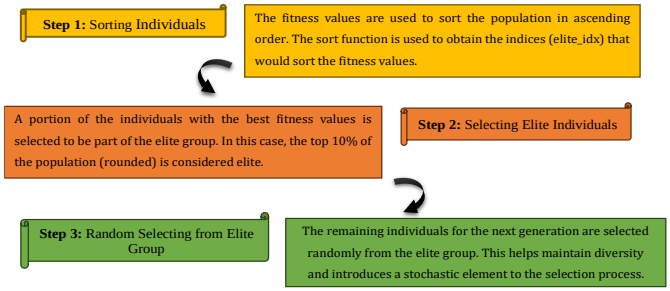


Fig. 1 *Example of Elitism operator [17]*

2.6 Crossover Operator

In this stage, regarding the TSP, the crossover operator is commonly employed to exchange genetic information between two parent individuals. This exchange results in the creation of one or more offspring, with genetic information representing the order or sequence of cities visited during the TSP tour [22]. Order crossover (OX) is the common crossover operator used for TSP and the example is as follows:

Let: Parent 1: [1,**3,2,4**,5]
Parent 2: [5,**2,3,1**,4]

Step 1:A subset of cities are randomly selected from one parent. For example, select the subset between **positions 2 and 4.**

Let: Subset of Parent 1: [3,2,4]
Subset of Parent 2: [2,3,1]

Step 2: Generate an offspring by keeping the same order of selected subsets from Parent 1 and Parent 2.

Offspring 1: [_,3,2,4, _]
Offspring 2: [_,2,3,1, _]

Step 3:Complete the remaining cities in the offspring by following the order of the cities from the other parent, ensuring that no cities are repeated.

Completed Offspring 1 based on Parent 2 order: [**5**,3,2,4,**1**]
Completed Offspring 2 based on Parent 1 order: [**4**,2,3,1,**5**]

The Order Crossover operator allows for the exchange of genetic algorithms between parents by keeping the certain order characteristics of each parent's path.

2.7 Mutation Operator

The mutation process involves the use of exchange techniques called swap mutation, where two random points (position of cities) in the chromosome are selected, and the elements corresponding to those points are exchanged with each other [23]. Fig. 2 provides an illustrative example of this mutation operator.

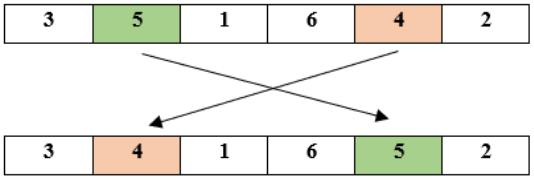


Fig. 2 *Example of mutation operator [19]*

Enhanced Knowledge in Sciences and Technology Vol. 4 No. 1 (2024) p. 171-182

2.8 Termination Condition

The decision to stop the algorithm is based on reaching the specified generation limit. Once this limit is reached, the algorithm is terminated. In this study, the generation limits used are 250, 500, and 750, with 10 repeated runs for each parameter tuning. Throughout the process, the distances between the tours and the computer's runtimes are carefully observed and recorded as the best distance along with the run time obtained respectively among the 10 repeated runs.

2.9 Flowchart of Research Framework

Fig. 3 is a visual representation of the completion process observed throughout this research. The figure illustrates outcomes, providing a comprehensive overview of the study's progression.

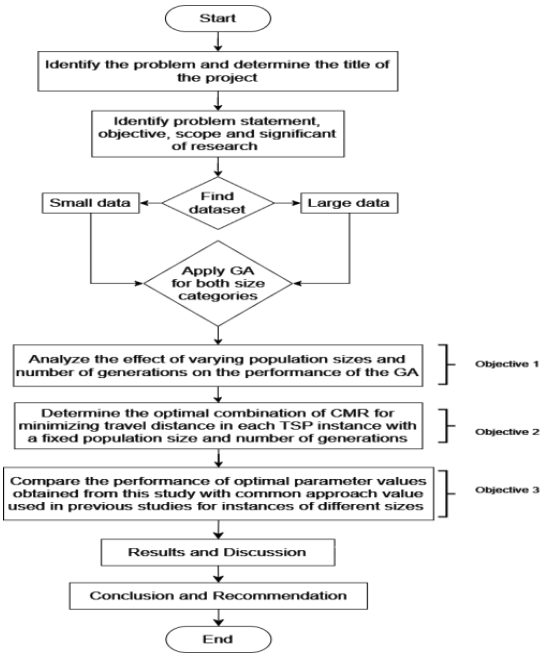


Fig. 3 *Flowchart of Research Framework*

2.10 Flowchart of Genetic Algorithm Process

Fig. 4 is a visual representation of the steps involved in the Genetic Algorithm (GA).

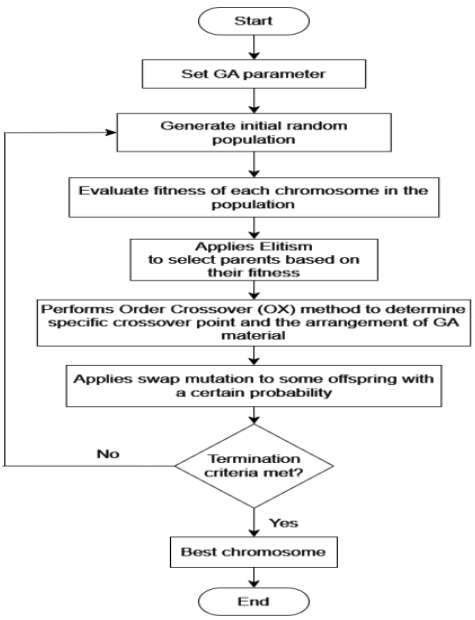


Fig. 4 *Flowchart of Genetic Algorithm (GA) [11]*

2.11 Parameter Tuning

Parameter tuning involves adjusting algorithm settings to improve its effectiveness in finding the best path. This adjustment includes using various genetic operators, such as different selection methods like Tournament and Ranking selection. Therefore, parameter tuning ensures that the algorithm performs well for different types and sizes of TSP problems. In this study, parameter tuning was performed by employing various parameter values in the GA with specific genetics operators such as Elitism selection, Order crossover (OX), and swap mutation. Table 4 shows the parameter values used in Genetic Algorithm.

Table 4 *Parameter values used in Genetic Algorithm*

Parameter	Value
Number of cities	42, 131, 1083, 2036
Population size	20, 40, 60, 80 and 100
Number of iterations	250, 500, 750
Selection method	Elitism
Crossover rate	0.7, 0.8 and 0.9
Mutation rate	0.01, 0.02, 0.4 and 0.5

2.12 Flowchart of Parameter Tuning

Fig. 5 is a visual representation of the steps involved in the experiment of parameter tuning.

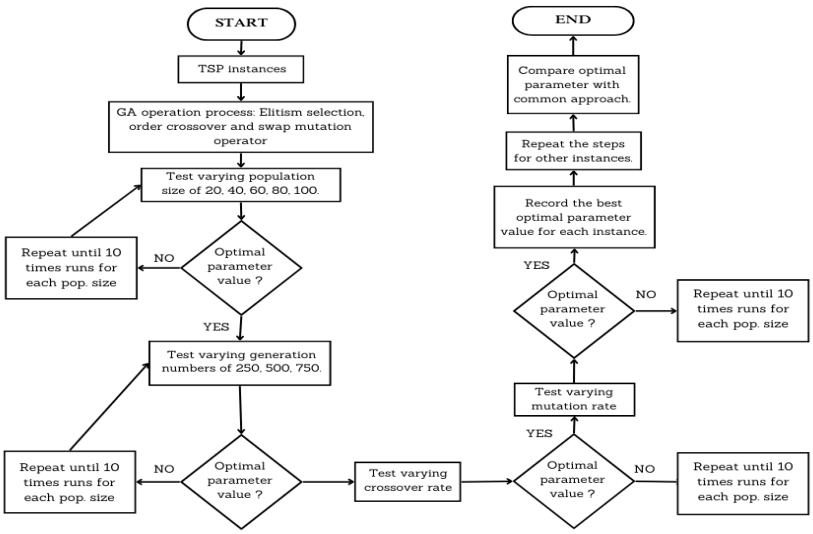


Fig. 5 *Flowchart of Parameter Tuning*

3. Results and Discussion

3.1 Experiments of TSP Instances

In this study, the experimental codes were developed using MATLAB R2023b software. The initial phase of the experiment involved utilizing the TSP instance of dantzig42, which comprises 42 cities. Overall, there are four sets of experiments and each experimental set consisted of four test phases. Then, the initial test applied the elitism selection method with fixed parameters of 750 generations, a crossover rate of 0.9, and a mutation rate of 0.01 with population size varied within a range from 20 to 100 in increments of 20. The results from 10 repeated runs for each population size were collected, and the lowest traveled distance was recorded as the best distance. Consequently, the population size of 100 which was associated with the shortest travel length value for these 42 cities will be held constant in the subsequent experiment. This experiment aims to analyze the impact of varying population sizes on the performance of the GA in solving the TSP across instances of different sizes.

Next, the testing phase proceeded by keeping other parameter values constant, while the number of generations varied specifically, at 250, 500, and 750. The recorded results followed the same steps as the previous experiment. As a result, 750 generations will be held constant in the subsequent experiment, which involves the

combination of crossover and mutation rates (CMR). The aim is to explore how changes in the number of generations influence the efficiency of the GA when solving the TSP across various instance sizes.

Then, the experiment was conducted by testing varying crossover rates at a fixed mutation rate of 0.01, and other parameter values were held constant as employed in the previous tests. The method of recording results remained the same. Thus, the crossover rate of 0.9 appears to be the optimal value for finding the shortest travel distance and computational time. The final test in this set of experiments involves observing varying mutation rates. Hence, the result obtained shows that the mutation rate of 0.01 is deemed to be the optimal parameter value for this particular instance of 42 cities. The experiment was repeated for the remaining three TSP instances consisting of 131, 1083, and 2036 cities.

3.2 Result of Parameter Tuning

The table below represents the results of near-optimal travel length and computational time for each parameter tuning in the previous experiments for different sizes of TSP instances.

3.3 Population size

Table 5 *Variations of population sizes at fixed generations = 750 and CMR (0.9, 0.01)*

Pop. Size	42 cities		131 cities		1083 cities		2036 cities	
	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)
20	2446.58	1.20	4077.75	1.87	81551.1	2.07	201279.9	4.84
40	2382.67	2.40	3584.55	3.63	79536	3.80	200675.9	6.09
60	2382.67	3.09	3390.69	4.82	80141.5	5.57	200421.2	9.44
80	2332.08	4.13	3467.6	5.92	79536	7.78	185771.6	11.61
100	**2215.75**	5.07	**3380.7**	6.83	**79688.2**	9.61	**185479.7**	16.94

The population size of 100 with 750 generations shows a consistent trend across all instances that have been tested. Based on the obtained results, a larger population size of 100 with 750 generations tends to produce better solutions compared to the smaller population size across different sizes of TSP instances used in this study. This is because it provides a more diverse set of potential solutions, allowing the algorithm to explore a broader portion of the solution space. Such diversity can aid in discovering a wider range of potential solutions, thereby increasing the chances of finding a global optimum. However, the time required tends to increase with population size, which is expected because the algorithm might require more computational time to converge.

3.4 Number of Generations

Table 6 *Variations of generation number at fixed pop. size = 100 and CMR (0.9, 0.01)*

Generation	42 cities		131 cities		1083 cities		2036 cities	
	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)
250	2860.99	1.79	3472.34	2.04	85939.9	3.34	194821	5.32
500	2382.67	3.59	3464.4	3.77	84622.9	6.64	193648.5	11.54
750	**2215.75**	5.07	**3337.46**	5.97	**79688.2**	9.61	**185479.7**	16.94

Similarly, a larger generation number also obtained the optimal solution of the best distance across different sizes of TSP instances. This is because a higher number of generations allows the algorithm more time for both exploration and exploitation. If the generation number is too small, it may lead the algorithm to converge early. When the algorithm reaches a point, it no longer explores new solutions and tends to settle on a suboptimal solution. Therefore, the balance between exploration and exploitation is crucial for finding optimal solutions.

3.5 Crossover Rate

Table 7 *Variations of crossover rate at fixed mutation = 0.01, pop. size = 100 and generation = 750*

Crossover	42 cities		131 cities		1083 cities		2036 cities	
	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)
0.7	2101.77	5.20	3494.49	6.38	85464.2	9.51	**184321.3**	15.27
0.8	2156.98	5.33	3464.31	6.07	85939.6	9.91	210900.3	15.31
0.9	**2078.89**	5.65	**3337.46**	5.97	**79688.2**	9.61	185479.7	16.94

Besides, the crossover rate varies across instances as shown in Table 7 where only the instance of 2036 cities use a crossover rate of 0.7 while other instances such as 42, 131, and 1083 cities have the same value crossover rate of 0.9. Generally, higher crossover rates such as 0.9 encourage more exploration of the solution space. In the case of 2036 cities, the crossover rate to achieve the best travel distance is 0.7, which is lower than in other instances.

3.6 Mutation Rate

Table 8 (a) *Variations of mutation rate at fixed crossover = 0.9, pop. size = 100 and generation = 750*

Mutation	42 cities		131 cities		1083	
	Distance	Time (s)	Distance	Time (s)	Distance	Time (s)
0.01	**2078.89**	5.65	**3337.46**	5.97	**79688.2**	9.61
0.02	2382.67	6.65	3636.9	5.79	80489.2	10.45
0.4	2860.99	6.86	4496.53	5.55	79769.7	9.97
0.5	2471.93	5.38	5178.99	5.36	80201.8	11.10

Table 8 (b) *Variations of mutation rate at fixed crossover = 0.7, pop. size = 100 and generation = 750*

Mutation	2036 cities	
	Distance	Time (s)
0.01	**184321.3**	**15.27**
0.02	205151.3	15.40
0.4	**184321.3**	16.87
0.5	190542.7	16.32

Furthermore, the mutation rate appears to have a similar value of 0.01 for all TSP instances indicating the likelihood of producing a favourable solution. The optimal combination value of crossover and mutation rates (CMR) was determined for minimizing travel distance at a fixed population size of 100 and 750 generations which is CMR (0.9, 0.01) for TSP instances of 42, 131, and 1083 cities while CMR (0.7, 0.01) was found to be optimal for 2036 cities. The difference in crossover value may be attributed to the importance of maintaining diversity in the population, as it is crucial to avoid premature convergence. Hence, a crossover rate of 0.7 may strike a better balance by allowing for sufficient exploration and preventing the loss of diversity, ultimately leading to the discovery of better solutions for this 2036 instance.

In contrast, smaller instances such as 42 cities with higher crossover and mutation rates (CMR = 0.9, 0.01) may benefit from a more explorative approach to efficiently search the solution space and find optimal solutions since it is a small dataset. Additionally, GA can be effective for solving large-size TSP instances such as 1083 and 2036 cities, but their performance depends on various factors, including the specific characteristics of the problem and the chosen algorithm parameters.

3.7 Summary of Optimal Parameter Values

Table 9 *Summary of optimal values across TSP instances using GA*

Parameter	42 cities	131 cities	1083	2036 cities
Population size	100	100	100	100
Number of generations	750	750	750	750
Crossover rate	0.9	0.9	0.9	**0.7**
Mutation rate	0.01	0.01	0.01	0.01

Table 9 shows a summary of optimal parameter values obtained from the experiment. Note that the experiment was conducted by taking the ten readings of the travel length when testing each parameter for population size, number of generations, crossover rate, and mutation rate. Then, the best distance obtained, which is the shortest tour is selected from those ten values and recorded in the table for each parameter being tested.

To further explore the influence of parameter choices, an additional experiment was conducted. This experiment involved comparing the results obtained using the common approach of a crossover rate of 0.9 and a mutation rate of 0.03, shedding light on how this standard configuration compares with the varying parameter settings observed in the initial experiments.

Enhanced Knowledge in Sciences and Technology Vol. 4 No. 1 (2024) p. 171-182

3.8 Comparison of Common Approach with Optimal Parameter

As shown in Table 10, a common approach is labeled as CMR 1 (0.9, 0.03), while the optimal approach, with two set values which are (0.9, 0.01) and (0.7, 0.01), is labeled as CMR 2.

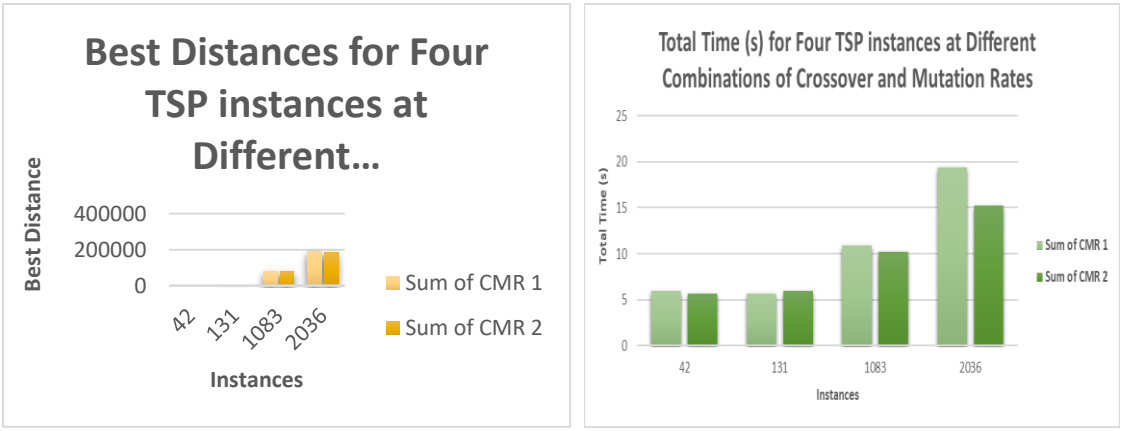
Table 10 *Parameter configurations for various TSP instances*

	Instances	Value (Crossover, Mutation)	Note
CMR 1	42 131 1083 2036	(0.9, 0.03)	Common Approach [11]
CMR 2	42 131 1083	(0.9, 0.01)	Optimal Parameter
	2036	(0.7, 0.01)	

Another experiment was conducted, and Table 11 represents the comparison of the results obtained for CMR 1 and CMR 2.

Table 11 *Comparison of the results obtained by different combinations of CMR*

Instance	CMR 1		CMR 2	
	Best Distance	Total Time (s)	Best Distance	Total Time (s)
42	2471.93	5.94	**2078.89**	**5.65**
131	**3323.19**	**5.68**	3337.46	5.97
1083	79542.19	10.91	**79535.99**	**10.20**
2036	190542.65	19.34	**184321.31**	**15.27**



(a) **(b)**

Fig. 6 *(a) Best distance of CMR; (b) Total time (s) of CMR*

Based on Fig. 6 (a), the analysis reveals minimal variation in the best distances achieved by CMR 1 and CMR 2 as illustrated in the graph. Nevertheless, CMR 2 (0.9, 0.01) consistently exhibits superior performance over CMR 1 (0.9, 0.03), especially in minimizing travel distance and total time, with significant advantages observed in instances of 42 and 1083 cities. However, this performance difference is not uniform across all instances, where CMR 1 has a slight advantage in an instance of 131 cities. This variation may be attributed to the specific characteristics of the instance, including potential differences in the geometrical arrangement of cities, the distance between each city, and the complexity of the solution space. Therefore, the specific arrangement of cities or the nature of the TSP instance of 131 might favour the exploration strategy employed by CMR 1. Besides, this difference also could be attributed to algorithm sensitivity, where the chosen operators, such as Order Crossover (OX) and swap mutation, can be highly sensitive to the choice of parameters. Thus, CMR 1 may be more suitable for the characteristics of this instance.

Furthermore, in the case of instance 2036, CMR 2 (crossover rate: 0.7, mutation rate: 0.01) outperformed CMR 1 (crossover rate: 0.9, mutation rate: 0.03) in both achieving a lower travel distance (Figure 6 (a)) and completing the optimization process in less time (Figure 6 (b)). This implies that, for this specific instance, the combination parameters of CMR 2 prove more effective than those of CMR 1. A lower crossover rate, as observed in CMR 2, may be attributed to the importance of maintaining diversity in the population, as it is crucial to avoid premature convergence. Hence, a crossover rate of 0.7 may strike a better balance by allowing for sufficient exploration and preventing the loss of diversity, ultimately leading to the discovery of better solutions for the TSP with 2036 cities. Conversely, a lower mutation rate can contribute to a more focused and refined exploitation, potentially preventing premature convergence to suboptimal solutions. It allows the algorithm to exploit different possibilities without introducing excessive randomness. This suggests that, for this specific problem size containing 2036 cities, striking a balance between exploration (lower crossover) and exploitation (lower mutation) is crucial, and CMR 2 strikes a better balance for this instance compared to CMR 1.

4. Conclusion

This paper analyzes how parameter tuning affects the performance of a GA to obtain the lowest travel length and computational time across four TSP instances that have not been used in previous research. The GA operators used in this study are Elitism selection, order crossover, and swap mutation operators. Four sets of experiments were conducted, with each set consisting of four phases of testing. The experiments were repeated for the remaining TSP instances.

The first objective has been successfully observed in Sections 3.2.1 and 3.2.2 which aim to study the effect of varying population sizes and number of generations on the performance of the GA in solving the TSP across various instance sizes. The results indicate that a population size of 100 was found to yield the best distance, indicating the lowest travel length across TSP instances. This is because a larger population size increases the diversity of solutions within a population. Despite this, a small population size may lead the algorithm to converge prematurely to suboptimal solutions. Next, each TSP instance being tested shows that the optimal number of generations yields the best distance at 750 generations as the algorithm has more opportunities to evolve and improve the solutions iteratively with more generations.

Secondly, the optimal combination value of crossover and mutation rates (CMR) was determined in Section 3.3 for minimizing travel distance at a fixed population size of 100 and 750 generations which is CMR (0.9, 0.01) for TSP instances of 42, 131, and 1083 cities while CMR (0.7, 0.01) was found to be optimal values for 2036 cities. The difference in crossover value may be attributed to the importance of maintaining diversity in the population, as it is crucial to avoid premature convergence. Hence, a crossover rate of 0.7 may strike a better balance by allowing for sufficient exploration and preventing the loss of diversity, ultimately leading to the discovery of better solutions for the 2036 instance.

Lastly, the performance comparison between optimal parameter values with a common approach is observed in Section 3.4. The results indicate that the optimal approach of CMR 2 (0.9, 0.01, and 0.7, 0.01) tends to outperform the common approach used of CMR 1 (0.9, 0.03) in terms of minimizing travel distance and computational time in this study. So, the results can be used to verify the hypothesis that a high crossover rate contributes to an increased accuracy score of the algorithm with a lower mutation rate (0.9, 0.01), while a reduced crossover rate may lead to an early convergence of the algorithm, particularly as large instances demonstrate the capability to identify the optimal travel distance. Hence, it may strike a better balance by allowing for sufficient exploration when a lower crossover is combined with lower mutation rates (0.7, 0.01) for the large instances of 2036 cities. Moreover, raising the mutation rate from 0.01 to 0.03 does not necessarily contribute to finding the best solution. Thus, the combination of CMR 2 contributes to better solutions in the TSP instances under study compared to CMR 1.

In future research, the paper may be extended by conducting comparative analyses between GA and other optimization methods such as Particle Swarm Optimization (PSO), Simulated Annealing (SA), or Ant Colony Optimization (ACO). Furthermore, hybrid approaches that combine the strengths of different optimization methods could be investigated to determine whether they yield improved results. Comparative studies across a diverse set of optimization algorithms and hybrid approaches would guide practitioners in selecting the most suitable method for specific problem instances.

Acknowledgement

The authors would thank the Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

The authors declare that there is no conflict of interest associated with the publication of this paper.

Author Contribution

The authors confirm the contribution as follows: ***study conception and design, data collection, analysis and interpretation of results, and manuscript preparation****: Nor Ain Shafiqah binti Abdullah;* ***supervision and review, editorial oversight, and proofreading****: Siti Noor Asyikin binti Mohd Razali.*

References

[1] Gupta, S., Abderazek, H., Yıldız, B. S., Yildiz, A. R., Mirjalili, S., & Sait, S. M. (2021). Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. *Expert Systems with Applications*, *183*, 115351. https://doi.org/10.1016/j.eswa.2021.115351

[2] Zheng, K., Zhang, Z., & Song, B. (2020). Retracted: E-commerce logistics distribution mode in big-data context: A case analysis of JD.COM. *Industrial Marketing Management*, *86*, 154–162.

[3] Antosiewicz, M., Koloch, G., & Kamiński, B. (2013). Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed. *Journal of Theoretical and Applied Computer Science*, *7*(1), 46-55.

[4] Mondal, M., & Srivastava, D. (2023). A genetic algorithm-based approach to solve a new time-limited travelling salesman problem. *International Journal of Distributed Systems and Technologies*, *14*(2), 1–14.

[5] Deng, Y., Liu, Y., & Zhou, D. (2015). An improved genetic algorithm with initial population strategy for symmetric TSP. *Mathematical Problems in Engineering*, *2015*, 1–6. https://doi.org/10.1155/2015/212794

[6] Singh, D. R., Singh, M. K., Singh, T., & Prasad, R. (2018). Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation. *Computación y Sistemas*, *22*(2), 491-503. https://doi.org/10.13053/cys-22-2-2956

[7] Hacizade, U., & Kaya, I. (2018). GA based traveling salesman problem solution and its application to transport routes optimization. *IFAC-PapersOnLine*, *51*(30), 620–625.

[8] Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., & Prasath, V. B. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, *10*(12), 390.

[9] Farzadnia, F., & Lysgaard, J. (2021). Solving the service-oriented single-route school bus routing problem: Exact and Heuristic Solutions. *EURO Journal on Transportation and Logistics*, *10*, 100054. https://doi.org/10.1016/j.ejtl.2021.100054

[10] Halim, A. H., & Ismail, I. (2019). Combinatorial optimization: Comparison of heuristic algorithms in travelling salesman problem. *Archives of Computational Methods in Engineering*, *26*(2), 367–380.

[11] Fu, C., Zhang, L., Wang, X., & Qiao, L. (2018). Solving TSP problem with improved genetic algorithm. *AIP Conference Proceedings*. 1967(1). https://doi.org/10.1063/1.5039131

[12] Ansari, A. Q., Ibraheem, & Katiyar, S. (2015). Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS. *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions, WCI 2015*, 68(2), 1-5.

[13] Donbosco, I. S., & Chakraborty, U. K. (2021). Comparing performance of evolutionary algorithms - A travelling salesman perspective. *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*.

[14] Loo, K., & Asrah, N. M. (2022). Survey on Customer Satisfaction Towards Courier Services in Johor. *Enhanced Knowledge in Sciences and Technology*, *2*(2), 186-196.

[15] Zulkiflee, N. F., & Rusiman, M. S. (2021). Heart Disease Prediction Using Logistic Regression. *Enhanced Knowledge in Sciences and Technology*, *1*(2), 177-184.

[16] Lim, L. S. Y. S., & Ismail, T. D. I. B. (2022). A Study On Volatility and Tail Risk of Small-Cap Companies in Comparison to Big-Cap Companies. *Enhanced Knowledge in Sciences and Technology*, *2*(1), 241-249.

[17] Haron, N. A. A., & Kamardan, M. G. (2021). Queuing system of a busy restaurant using simulation software. *Enhanced Knowledge in Sciences and Technology*, *1*(2), 66-71.

[18] Al-Khatib, R. E. M., Al-Betar, M. A., Awadallah, M. A., Nahar, K. M., Shquier, M. M. A., Manasrah, A. M., & Doumi, A. B. (2019). MGA-TSP: modernised genetic algorithm for the travelling salesman problem. *International Journal of Reasoning-based Intelligent Systems*, *11*(3), 215-226.

[19] Katoch, S., Chauhan, S. S., & Kumar, V. (2020). A review on Genetic Algorithm: Past, present, and future. *Multimedia Tools and Applications*, *80*(5), 8091–8126.

[20] Singh, A. N., Mrudula, J., Pandey, R., & Das, S. (2021). A comparative study of four genetic algorithm-based crossover operators for solving travelling salesman problem. *Intelligent Algorithms for Analysis and Control of Dynamical Systems*, 33-40. https://doi.org/10.1007/978-981-15-8045-1_4

[21] Romero-Hdz, J., Aranda, S., Toledo-Ramirez, G., Segura, J., & Saha, B. (2016). An Elitism Based Genetic Algorithm for Welding Sequence Optimization to Reduce Deformation. *Res. Comput. Sci.*, *121*, 17-36.

[22] Bao, C., Yang, Q., Gao, X.-D., & Lu, Z.-Y. (2022). Genetic algorithm with adapted crossover operators for multiple traveling salesmen problem with visiting constraints. *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3033–3039. https://doi.org/10.1109/smc53654.2022.9945175

[23] Maier, H. R., Razavi, S., Kapelan, Z., Matott, L. S., Kasprzyk, J., & Tolson, B. A. (2019). Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental modelling & software*, *114*, 195-213. https://doi.org/10.1016/j.envsoft.2018.11.018