

Optimization of Adaptive Genetic Algorithm Parameters in Traveling Salesman Problem

I Kayan Herdiana ^{1)*}, I Made Candiasa ²⁾, Gede Indrawan ³⁾

¹⁾²⁾³⁾ Universitas Pendidikan Ganesha, Indonesia

¹⁾herdikayan@gmail.com, ²⁾candiasa@undiksha.ac.id, ³⁾gindrawan@undiksha.ac.id

ABSTRACT

The TSP problem is where a seller visits multiple destinations simultaneously, and they are only allowed to visit once. This TSP aims to shorten the shortest distance, thereby minimizing time and cost. There are various methods to solve the TSP problem, including greedy algorithm, brute force algorithm, hill-climbing method, ant algorithm, and genetic algorithm. Each process in a genetic algorithm is affected by several parameters, including population size, maximum generation, crossover rate, and mutation rate. The purpose of this study is the application of genetic algorithms to the traveling salesman problem, calculating the ultimate effect of generation, chromosome number, crossover rate, and mutation rate on the optimal genetic algorithm, to how the effects of adaptive genetic algorithm parameters on genetic algorithm results. Based on the results obtained from research and testing, the four parameters of the genetic algorithm are positively correlated with fitness results while negatively correlated with execution time performance, where each adaptive parameter applied provides more optimal fitness results than static parameters. The four adaptive parameters are used together to give optimal results, both fitness which reaches 1.0%, and time reaches 38.7%.

Keywords: Fitness, Genetic Algorithm, Optimization, Performance, TSP.

INTRODUCTION

The TSP problem is where a seller visits multiple destinations simultaneously and is only allowed to visit once. This TSP aims to shorten the shortest distance, minimizing time and cost. TSP is a classic problem that occurs frequently, but its solution requires complex algorithms, mainly when many places are visited. For routes with less space, you can use a brute force algorithm. But when the points reach 20, the number of combinations of the Hamiltonian circuit is 6×1016 (Nam & Maslov, 2019). A Hamiltonian cycle is a circuit that traverses each vertex in the graph only once, except for the start and end points, which are traversed twice (Utomo, 2017). In TSP, a bidirectional graph holds, where the distance between A and B is not the same as the distance between B and A. Any vertex can go to any other vertex and can pass the same edge twice. This is in contrast to an Euler circuit, which is a circuit that traverses each edge of a graph only once, even if it crosses a vertex more than once (Huang, Xu, & Zhang, 2020). The absence of standard rules in setting the value of these parameters becomes a difficulty in the use of evolutionary algorithms to solve problems (Muzid, 2020).

TSP is one of the fundamental combinatorial optimization problems. TSP has many applications in operational research (Zhang, Tong, Xu, & Lin, 2016). Genetic algorithms (GA) use natural selection, a process known as evolution (Ginatra & Anandita, 2019). In any evolutionary process, individuals or genes in chromosomes are constantly being altered to suit their environment. The main steps of the algorithm are the selection, crossover, and mutation process of each generation. GA is often combined with other methods for more optimal results, such as Dynamic Artificial Chromosomes (Mursalim, Purwanto, & Soeleman, 2021), Simulated Annealing (Amine, 2019), Partially Mapped Crossover (Hardi, Zarlis, Effendi, & Lydia, 2020), and Shared Neighbors (Cao, Chen, & Wang, 2022). A genetic algorithm is a solution method that can be classified as a heuristic. Heuristics are approximate or probabilistic algorithms for the solution you are trying to find (Gamrath, Berthold, Heinz, & Winkler, 2019). The final result of a genetic algorithm is a collection of solutions represented in the form of chromosomes. As a final solution, use one of the most evolved chromosomes with the highest fitness. The answer in a genetic algorithm is not always the same in every experiment because the actions at each step are random.

Each process in a genetic algorithm is affected by several parameters, including population size, maximum generation, crossover rate, and mutation rate (Villalba Matamoros & Kumral, 2019). In each generation, chromosomes undergo evolution to achieve optimal outcomes. The magnitude of change is determined by

* Corresponding author



predetermined probability parameters, namely the crossover and mutation rates. Selecting the size of each parameter affects the performance of the genetic algorithm, including execution time and memory consumption. With GA, optimal fitness does not have to be at the end of a generation. Fast and accurate output is significant in TSP problems to derive the desired route based on selected points in real-time. Therefore, it is necessary to determine each parameter to achieve the best results correctly.

This research is urgent to solve the TSP problem, where there are no standard rules in determining genetic algorithm parameters such as population size, number of generations, crossover rate, and mutation rate. The genetic algorithm parameters that affect the performance of the genetic algorithm in this study are execution time and memory usage. The application of GA in TSP optimization prioritizes the accuracy of the results and the search time. So that if this research is not carried out properly, TSP problems will still experience deficiencies in the accuracy of results and search time. Their implementation will affect companies implementing CSR, such as waste of resources, financial losses, and other operational risks. The purpose of this study is the application of GA to the traveling salesman problem, calculating the maximum effect of generation, chromosome number, crossover rate, and mutation rate on the optimal genetic algorithm, to how the effects of adaptive genetic algorithm parameters on genetic algorithm results.

LITERATURE REVIEW

The Traveling Salesman Problem (TSP) belongs to the class of hard NP (non-deterministic polynomial-time) and usually uses heuristics to find a solution (Srinivasan, Satyajit, Behera, & Panigrahi, 2018). The main issue in the TSP case is how the seller can design his itinerary to visit multiple cities, where the distance from one city to another is known to reach a minimum total distance, and the seller can only visit that city once. There are various methods to solve the TSP problem, including the greedy algorithm (Wu & Fu, 2020), brute force algorithm (Violina, 2021), hill-climbing method (Fronita, Gernowo, & Gunawan, 2018), ant algorithm (Chen, Tan, Qian, & Chen, 2018), and genetic algorithm. The problem faced in TSP is how to find the minimum total distance. Solving this problem is not easy because in TSP, there is a search space for a set of permutations of several cities, so TSP is known as a non-Polynomial problem. A simple description of the TSP problem is shown in Figure 1. Each city in Figure 2. 1 has coordinates (x, y) so that the distance between the two cities can be determined using this formula

$$\text{The } d_{(ij)} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

x_i = x coordinates of city i, x_j = x coordinates of city j, y_i = y coordinates of city i and y_j = y coordinates of city j. After the distance of each city is known, it will look for the path of the path with the minimum distance that will be passed to return to the initial city again.

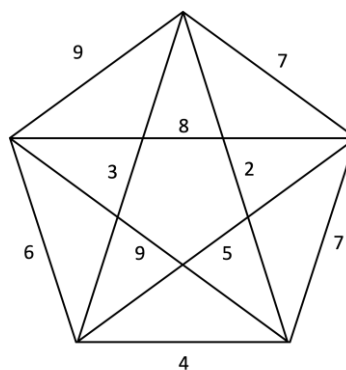


Fig. 1 Position of Cities to be Passed

Genetic algorithm, as a branch of evolutionary algorithm, is a method for solving value search in optimization problems, namely nonlinear problems (Slowik & Kwasnicka, 2020). The history of GA was first proposed by John Holland of the University of Michigan in New York, the USA, in the early 1970s. The genetic algorithm differs from traditional convergence techniques in that it is more deterministic (Wang, Lang, & Mao, 2021). GA applies the mechanisms of natural selection and genetics, so the terms in GA correspond to the terms of natural selection and

* Corresponding author



genetics. In general, the process of solving problems using GA is as follows. Initialization is a process carried out to generate a new set of solutions randomly into a population. The population formed is adjusted to the problem to be solved. At this stage, the population size must be determined based on the number of individuals who want to be accommodated. Crossover is the process of crossbreeding between two randomly selected individuals by choosing the cut point first. This crossover process aims to obtain offspring that will increase the variety of individuals in the population. The mutation is a process of randomly changing bits of chromosomes by choosing 2 points at random then, between the two points, the genes will be swapped. The mutation process serves to make the individuals in the population more varied. Evaluation is the process of collecting all individuals in the population.

In this process, the fitness value of each individual is searched. The fitness value obtained will represent the solution quality from the individuals formed. Selection is selecting the best individual from all alternative solutions to take several individuals with the expected fitness value. Several approaches have been proposed to adapt the parameter values of Ant Colony Optimization (ACO), which includes ACSGA-TSP, by Pilat and White, and Near parameter free ACS, by Randall. The parameter values in ACS were optimized using a genetic algorithm. The idea was to run a GA on top of the ACS to optimize its parameter values. The optimized parameters were q_0 , the parameter determining whether the greedy or probabilistic selection is adopted, ρ , the local pheromone updating factor, and β , the relative importance of the visibility heuristic.

The adaptive generation is related to the fitness obtained. In the case of the best fitness scheduling, it will be worth 1 so that this occurs when all schedules do not conflict. This condition makes it possible to stop the process even though it has not yet reached the last generation. In contrast to the TSP case, there is no condition with a specific fitness value to achieve optimal results where the result is the shortest distance. This condition forces the algorithm to iterate until the last generation and choose the best fitness from all these generations. In this case, adaptive generation is done by determining the value of MFH (Maximum of Fitness History), the smallest maximum fitness value stored in the iteration process. The experiment was carried out by giving the value of MFH = 50. It stores the smallest fitness value in each generation. If the number of fitness stored in the history is more than many MFH values, then the first history will be deleted. If the amount of wellness in the account is as many as MFH values and all values are the same, the iteration is stopped even though it has not reached the last iteration.

Adaptive Population Size (PS) is done by reducing the number of chromosomes in each generation until it reaches a certain number of Population Sizes called MPS (Minimum of Population Size). The PS value will reach its minimum when it reaches the last generation. Determine the PS at a certain age by using the following formula.

$$nPS = MPS + \frac{(G - nG)}{G} \times (PS - MPS) \quad (2)$$

Where nPS is the population size in the nth generation, MPS is the minimum of population size, G is the generation parameter, nG is the n-th generation. Reducing the population size is done by eliminating the chromosomes with the smallest fitness (longest distance traveled).

The Adaptive Crossover Rate value will be adjusted dynamically using the Decreasing of High Crossover (DHC) method (Hasannat: 2019). The DHC method can be formulated as follows.

$$CR = 1 - \left(\frac{LG}{Gn}\right) \text{ where } G = [1, 2, 3, \dots, Gn] \quad (3)$$

Where CR is the crossover rate, Gn is the total generation value. LG is a generation level number (current generation). In this case, based on Figure 2 and Figure 3, the writer limits the minimum CR with the following formula.

$$CR = 1 - \left(\frac{LG}{Gn}\right) * (1 - \min CR) \quad (4)$$

Where $\min CR$ is the minimum of crossover rate. The $\min CR$ value used in the study was 20 percent.

The Adaptive Mutation Rate value will be adjusted dynamically using the Increasing of Low Mutation (ILM) method. ILM method can be formulated as follows.

* Corresponding author



$$MR = \frac{LG}{Gn} \text{ where } G = [1, 2, 3, \dots, Gn] \quad (5)$$

Where MR is the mutation rate, Gn is the total generation value. LG is a generation level number (current generation). In this case, based on Figure 2 and Figure 3 where the performance is lacking for mutations and stability has been achieved when the MR is 30%, the authors limit the maximum MR with the following formula.

$$MR = \frac{LG}{Gn} * maxMR \quad (6)$$

Where $maxMR$ is the maximum Mutation Rate. The $maxMR$ value used in the study was 50 percent.

METHOD

During the literature phase, researchers examine scientific information and knowledge, including research related to ongoing research. Researchers obtain study data from various sources, including the Internet, during the data collection phase. In this study, the number of points or locations used was 15 points. They use the Google Map API to calculate the distance of each end using a case study of the area of schools in Denpasar City. During the method analysis phase, the researchers used multiple methods for each stage of the genetic algorithm. The processes used are the input parameters of the genetic algorithm, the selection process, and the transition to mutation process. In the system testing stage, the researchers tested the system in three phases: algorithm testing, algorithm parameter testing, Pearson correlation testing, and adaptive parameter testing. Conclusions are drawn by analyzing the results of system tests. By testing the algorithm's parameters, the author can determine the optimal parameter range of the genetic algorithm in the case of the traveling salesman problem. The Pearson correlation test produces a number showing how much the input variables affect the algorithm's output. The research flowchart can be seen in Figure 2.

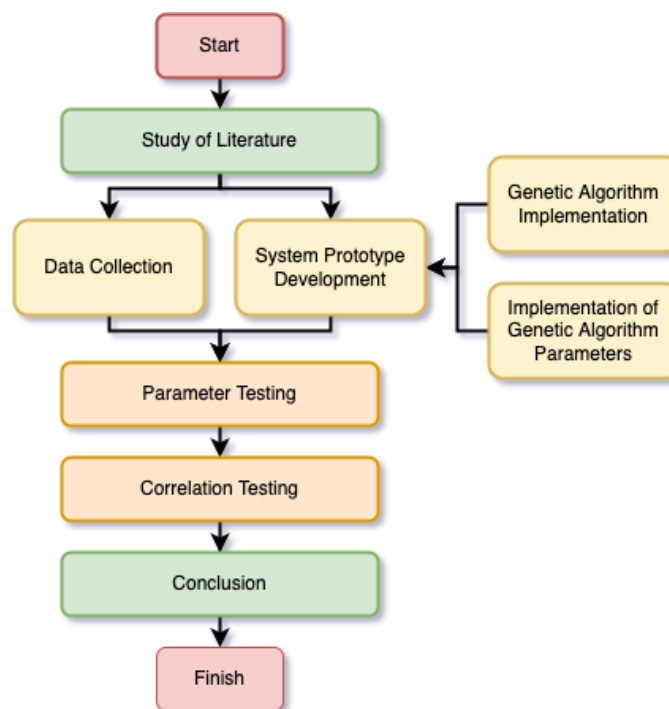


Fig. 2 Research Flowchart

RESULT

Genetic Algorithm Parameter Testing

This test is done by changing the value of each parameter of the genetic algorithm, namely Generation (G), Population Size (PS), Crossover Rate (CR), and Mutation Rate (MR) to find the shortest distance solution from the TSP case with 15 points. The test results of these parameters will be correlated with Fitness (shortest distance) and Performance (execution time).

* Corresponding author

Test Grouping

This test is grouped into 4 groups with 200 tests each as follows. Generation test tests AG with random G values between 1 to 1000 and PS (100), CR (90%), MR (3%). Population Size Test, tested AG with PS values randomly between 2 and 100 and G values (1000), CR (90%), MR (3%). The minimum PS value is set to 2 because the crossover process requires at least 2 parent chromosomes. Crossover Rate Test, testing AG with CR values randomly between 0 to 100 percent and G values (1000), PS (100), MR (3%). Mutation Rate Test, tests AG with MR values randomly between 0 to 100 percent and G values (1000), PS (100), CR (90%).

Test result

Based on the 4 test groups, the influence of parameters on fitness and performance was obtained as follows. Figure 2 illustrates the relationship between the four AG parameters on AG performance. Version (execution time) uses a range of 0-1, where 0 is the lowest performance (longest execution time), and 1 is the highest performance (fastest execution time). The AG parameter uses a range of 0 to 100 where 0 is the smallest parameter and 100 is the most significant parameter. Based on Figure 3, information is obtained that the larger the AG parameter, the lower the AG's performance or, the longer the execution time. The correlation of each parameter to performance (execution time) is as follows, where Population Size is negatively correlated with performance with a correlation of 0.996 (high). Generation is negatively correlated with performance with a correlation of 0.992 (high). Mutation Rate is negatively correlated with execution with a correlation of 0.767 (high enough). Crossover Rate is negatively correlated with performance with a correlation of 0.975 (high).

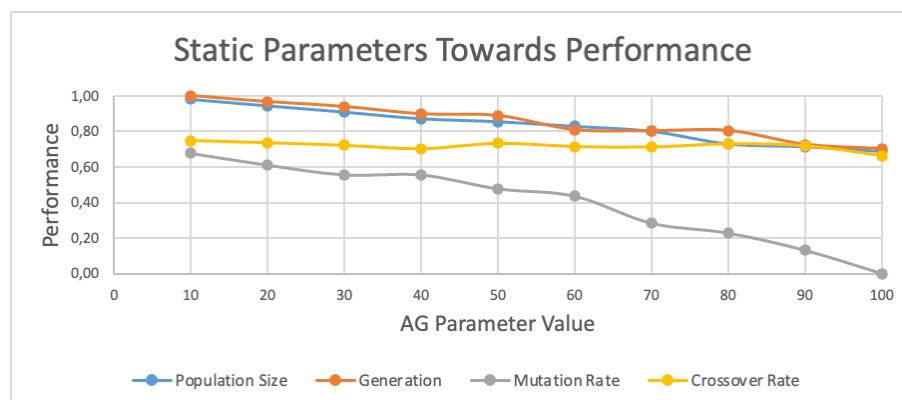


Fig. 3 Comparison Graph of AG Parameters Value Towards Performance

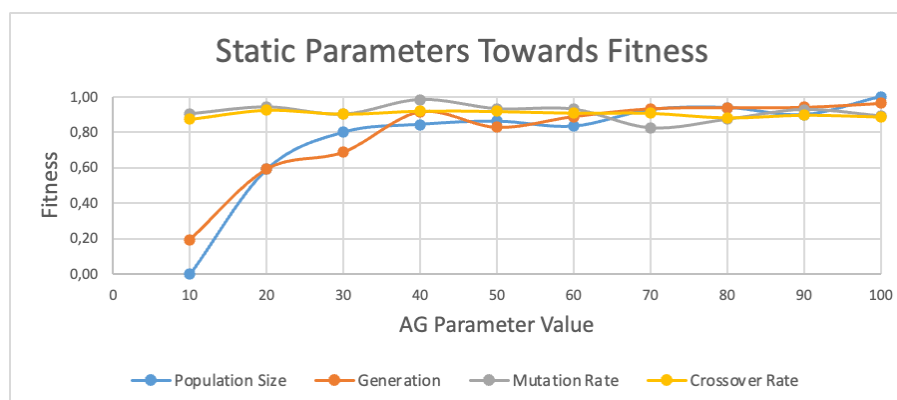


Fig. 4 Comparison Graph of AG Parameters Value Towards Fitness

Figure 4 illustrates the relationship between the four ag parameters on fitness (shortest distance). Fitness uses a range of 0-1, where 0 is the lowest fitness (longest distance), and 1 is the highest fitness (shortest distance). The AG parameter uses a range of 0 to 100 where 0 is the smallest parameter, and 100 is the largest parameter. Based on

* Corresponding author



Figure 3, information is obtained that the greater the AG parameter, the better the AG fitness or, the shorter the distance. The correlation of each parameter to fitness is as follows. Population size positively correlates with fitness with a correlation of 0.783 (high enough). Generation is positively correlated with fitness with a correlation of 0.385 (low). Mutation Rate is positively associated with fitness with a correlation of 0.671 (high enough). Crossover Rate is positively associated with fitness with a correlation of 0.522 (relatively low).

Adaptive Parameter Test Basis

The graph of the test results will be used as a reference for the next test, both the size of the parameter and the adaptive method used. These results can be described as follows. Parameter generation and population size achieve the highest performance if the value is getting bigger. The initial for these two parameters uses the highest value from the experiment, which is 1000 for generation and 100 for population size. The crossover rate parameter is relatively stable when tested for numbers 0 to 100 in terms of performance, and the fitness stability is at 50 and above. These results indicate that crossover can be used with large values. The mutation rate parameter produces a small performance, both from 0-100. But MR has reached a stable fitness when using values at 30 and above. These results show that mutations can be used with small weights.

Adaptive Parameter Test

The second test is done by changing the genetic algorithm process's parameter values in each generation. Change the parameter value of each parameter of the genetic algorithm, namely Generation (G), Population Size (PS), Crossover Rate (CR), and Mutation Rate (MR), by increasing the maximum until a value or reducing the minimum until value.

Adaptive Parameter Method

Adaptive Test Results

This test is divided into six groups: normal AG, adaptive generation, adaptive population size, adaptive crossover rate, adaptive mutation rate, and a combination of the four adaptive parameters. Each group was tested 100 times, then the average fitness and performance were taken. The results of this test can be seen in the table below:

Table 1
Adaptive Parameter Test Results

	Fitness (km)	Execution Time (seconds)
Normal AG	43,37	4,88
Adaptive Generation	43,27	2,55
Adaptive Population Size	43,34	2,96
Adaptive Crossover	43,19	4,41
Adaptive Mutation	43,25	4,99
Adaptive All Params	42,94	2,99

The green numbers in Table 1 mean that the results are more optimal than normal AG, while the red ones mean less than normal AG. Using adaptive mutation parameters gives lower performance than normal AG even though it provides better fitness. This happens because seen from the MR value when using adaptive parameters from 0 to 50 percent; then the average is 25 percent, 25 percent higher than the normal AG, which is 3 percent. The deficiency in this adaptive mutation can be covered by combining adaptive parameters to produce a much more optimal fitness than normal ag and other adaptive parameters.

DISCUSSIONS

TSP (Travelling Salesman Problem) is a problem where a salesman visits several destinations, where each place is called only once and returns to its original position. A genetic algorithm is one of the methods used to solve TSP problems. Determining the proper parameters will make the algorithm work optimally. The first experiment was carried out by entering different parameter values 200 times for each parameter. The first experiment's results describe the correlation of the four parameters to the fitness and execution time of the algorithm. The four parameters positively correlate to fitness, where the more significant the parameter, the better the fitness (, the

* Corresponding author



smaller the distance traveled). Compared to time, the greater the parameter value, the lower the performance (, the more time it takes).

The second experiment was carried out by changing each algorithm parameter to be adaptive, where each generation of parameter values constantly changed. The population size parameter is reduced each age from 50 to 2 chromosomes. The generation parameter will stop when it reaches the minor fitness 50 times. Each generation's crossover rate parameter is reduced from 100% to 20%. The mutation rate parameter is added every generation from 0 to 50%. The results of this second experiment show that adaptive parameters provide optimal fitness results and optimal time.

CONCLUSION

Based on the results obtained from research and testing, it can be concluded as follows. The four parameters of the genetic algorithm with the TSP case study are positively correlated with fitness results while negatively correlated with execution time performance. Each adaptive parameter applied gives more optimal fitness results than the static parameters where population size is 0.1%, generation is 0.2%, crossover rate is 0.4%, and mutation rate is 0.3%. Each adaptive parameter applied gives optimal time performance results except for mutation rate where population size is 39.3%, generation 47.7%, crossover rate 9.6%, mutation rate -2.3%. The four adaptive parameters are applied together to give optimal results, both fitness which reaches 1.0%, and time reaches 38.7%.

REFERENCES

- Amine, K. (2019). Multiobjective Simulated Annealing: Principles and Algorithm Variants. *Advances in Operations Research*, 2019, 1–13. <https://doi.org/10.1155/2019/8134674>
- Cao, Z., Chen, H., & Wang, X. (2022). Spectral clustering based on the local similarity measure of shared neighbors. *ETRI Journal*, etrij.2021-0230. <https://doi.org/10.4218/etrij.2021-0230>
- Chen, H., Tan, G., Qian, G., & Chen, R. (2018). Ant Colony Optimization With Tabu Table to Solve TSP Problem. *2018 37th Chinese Control Conference (CCC)*, 2523–2527. <https://doi.org/10.23919/ChiCC.2018.8483278>
- Fronita, M., Gernowo, R., & Gunawan, V. (2018). Comparison of Genetic Algorithm and Hill-climbing for Shortest Path Optimization Mapping. *E3S Web of Conferences*, 31, 11017. <https://doi.org/10.1051/e3sconf/20183111017>
- Gamrath, G., Berthold, T., Heinz, S., & Winkler, M. (2019). Structure-driven fix-and-propagate heuristics for mixed integer programming. *Mathematical Programming Computation*, 11(4), 675–702. <https://doi.org/10.1007/s12532-019-00159-1>
- Ginantra, N. L. W. S. R., & Anandita, I. B. G. (2019). Implementasi Algoritma Genetika Berbasis Web pada Sistem Penjadwalan Mengajar di SMK Dwijendra Denpasar. *Jurnal Teknologi Informasi Dan Komputer*, 5(1), 130–138.
- Hardi, S., Zarlis, M., Effendi, S., & Lydia, M. S. (2020). Taxonomy Genetic Algorithm For Implementation Partially Mapped Crossover In Travelling Salesman Problem. *Journal of Physics: Conference Series*, 1641(1), 012104. <https://doi.org/10.1088/1742-6596/1641/1/012104>
- Huang, X., Xu, C., & Zhang, L. (2020). An Efficient Algorithm for Optimizing the Test Path of Digital Microfluidic Biochips. *Journal of Electronic Testing*, 36(2), 205–218. <https://doi.org/10.1007/s10836-020-05865-6>
- Mursalam, Purwanto, & Soeleman, M. A. (2021). Penentuan Centroid Awal Pada Algoritma K-Means Dengan Dynamic Artificial Chromosomes Genetic Algorithm Untuk Tuberculosis Dataset. *Techno.Com*, 20(1), 97–108. <https://doi.org/10.33633/tc.v20i1.4230>
- Muzid, S. (2020). An Adaptive Approach to Controlling Parameters and Population Size of Evolutionary Algorithm. *Journal of Physics: Conference Series*, 1430(1), 012048. <https://doi.org/10.1088/1742-6596/1430/1/012048>
- Nam, Y., & Maslov, D. (2019). Low-cost quantum circuits for classically intractable instances of the Hamiltonian dynamics simulation problem. *Npj Quantum Information*, 5(1), 44. <https://doi.org/10.1038/s41534-019-0152-0>
- Slowik, A., & Kwasnicka, H. (2020). Evolutionary Algorithms and Their Applications to Engineering Problems. *Neural Computing and Applications*, 32(16), 12363–12379. <https://doi.org/10.1007/s00521-020-04832-8>
- Srinivasan, K., Satyajit, S., Behera, B. K., & Panigrahi, P. K. (2018, May 28). *Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience*. arXiv. Retrieved from <http://arxiv.org/abs/1805.10928>

* Corresponding author



-
- Utomo, R. B. (2017). Sirkuit Hamilton dalam Permainan Congklak. *AdMathEdu : Jurnal Ilmiah Pendidikan Matematika, Ilmu Matematika dan Matematika Terapan*, 7(1), 39. <https://doi.org/10.12928/admathedu.v7i1.7400>
- Villalba Matamoros, M. E., & Kumral, M. (2019). Calibration of Genetic Algorithm Parameters for Mining-Related Optimization Problems. *Natural Resources Research*, 28(2), 443–456. <https://doi.org/10.1007/s11053-018-9395-2>
- Violina, S. (2021). Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP). *Turkish Journal of Computer and Mathematics Education*, 12(8), 1226–1229. <https://doi.org/10.17762/turcomat.v12i8.3031>
- Wang, H., Lang, X., & Mao, W. (2021). Voyage optimization combining genetic algorithm and dynamic programming for fuel/emissions reduction. *Transportation Research Part D: Transport and Environment*, 90, 102670. <https://doi.org/10.1016/j.trd.2020.102670>
- Wu, C., & Fu, X. (2020). An Agglomerative Greedy Brain Storm Optimization Algorithm for Solving the TSP. *IEEE Access*, 8, 201606–201621. <https://doi.org/10.1109/ACCESS.2020.3035899>
- Zhang, H., Tong, W., Xu, Y., & Lin, G. (2016). The Steiner Traveling Salesman Problem with Online Advanced Edge Blockages. *Computers & Operations Research*, 70, 26–38. <https://doi.org/10.1016/j.cor.2015.12.013>