GENETIC ALGORITHM-BASED EFFICIENT ROUTE PLANNING FOR TRAVELING SALESMAN PROBLEM

Andysah Putera Utama Siahaan
Faculty of Science and Technology, Universitas Pembangunan Panca Budi

Keywords:
artificial intelligence, TSP, shortest path, optimum

***Correspondence Address:**
andiesiahaan@gmail.com

Abstract: This paper proposes a solution to the Traveling Salesman Problem (TSP) using a Genetic Algorithm (GA) approach. The GA is a technique that imitates natural evolution to produce and improve candidate solutions for the TSP. The proposed algorithm improves the efficiency of route planning by using a unique crossover and mutation operator that lowers the chances of getting stuck in local optima. The algorithm was tested on benchmark datasets, and the results showed that it performs better than others existing GA methods in terms of finding the shortest route with the least number of generations. This proposed algorithm has promising potential applications in logistics and transportation planning. The genetic algorithm can generate populations in order to find the minimum value, which may not be the global minimum, but it can still provide an optimal solution. By utilizing artificial intelligence, specifically in the context of the Traveling Salesman Problem, the distance traveled can be optimized.

INTRODUCTION

Efficient route planning is a crucial problem in the field of logistics and transportation, and the traveling salesman problem (TSP) is one of the most challenging problems in this domain (Zhou et al., 2018). With the advent of artificial intelligence (AI), researchers have started exploring the use of AI techniques to solve the TSP. Genetic algorithms (GAs) have emerged as a popular technique to solve the TSP in the AI environment. GAs are a type of optimization algorithm inspired by the process of natural selection. These algorithms can efficiently explore large solution spaces and identify optimal or near-optimal solutions.

The use of GAs in the AI environment has several advantages. Firstly, GAs can be parallelized, which makes them suitable for large datasets (Siahaan, 2016a). Secondly, GAs are robust and can handle noisy or incomplete data. Thirdly, GAs can adapt to changing environments and can update their solutions based on new information. Efficient route planning using GAs in the AI environment has several practical

applications, such as optimizing the delivery routes of packages or finding the shortest route for a fleet of vehicles. Moreover, the use of GAs in the AI environment can lead to significant time and cost savings for logistics and transportation companies.

The genetic algorithm operates by generating unpredictable random values, making it impossible to determine the expected results. Each iteration of the algorithm yields a unique output, and although it cannot guarantee the minimum mileage for the Traveling Salesman Problem, it can produce an optimal global value. The accuracy of the output depends on the duration of the fitness search performed by the genetic algorithm, with longer search processes resulting in smaller error values (Deng et al., 2015). Utilizing this algorithm holds the potential to effectively address the Traveling Salesman Problem.

Therefore, the proposed title "Efficient Route Planning for the Traveling Salesman using Genetic Algorithm in artificial intelligence environment" highlights the need for efficient route planning using GAs in the AI environment and underscores the significance of this approach for logistics and transportation.

THEORIES

2.1 Artificial Intelligence

Artificial Intelligence (AI) is a branch of science that aims to develop systems capable of performing tasks that typically require human intelligence. These systems can think, learn, and act like humans or even surpass human abilities. AI technology includes various techniques and methods such as machine learning, neural networks, natural language processing, computer vision, and automated reasoning. With these techniques, systems can perform tasks such as facial recognition, language translation, data-driven decision making, or even autonomous driving (Siahaan et al., 2018).

AI has been widely applied in various fields such as healthcare, manufacturing, transportation, government, and others. For example, AI can assist doctors in diagnosing diseases, accelerate production in factories, improve safety and efficiency in transportation, and predict natural disasters.

Although AI has many benefits, its use also raises ethical and security issues such as data privacy and discriminatory decision-making. Therefore, the development and implementation of AI need to be done responsibly and ethically (Gabriel, 2016).

2.2 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a well-known and challenging combinatorial optimization problem in computer science that involves finding the shortest path or tour that visits each point or city in a given set of points or cities exactly once, and finally returns to the starting point. TSP is commonly encountered in various fields such as logistics, shipping, networking, and supply chain management. The difficulty of TSP lies in the fact that the number of possible solutions increases exponentially with the number of points or cities to be visited. For example, for 10 cities, there are over 3.6 million different possible tours. In practice, TSP cannot be solved exactly or optimally for large problem sizes (Tabatabaee, 2015).

To solve TSP, there are several approaches, such as exact and heuristic approaches. Exact approaches attempt to solve TSP by evaluating every possible solution explicitly, while heuristic approaches efficiently search for solutions by disregarding irrelevant solutions. One of the most common heuristic approaches used to solve TSP is the Genetic Algorithm.

The Genetic Algorithm is an optimization method based on the principle of natural evolution. This algorithm works by generating random solutions naturally, so the estimated results cannot be predicted (Borna & Hashemi, 2014). Each trial of the Genetic Algorithm produces different outputs. The algorithm is capable of providing optimal global results, but cannot guarantee the minimum distance traveled in TSP. The smallest value obtained depends on how long the optimal search with the Genetic Algorithm is performed. The longer the search, the smaller the error value (El Hassani et al., 2015).

Although the Genetic Algorithm can provide globally optimal solutions, in practice, it is often used to find suboptimal solutions for complex TSP problems. This algorithm can find solutions that are close enough to the optimal solution in a short amount of time. TSP is a challenging problem that is commonly used in computer science to test new optimization algorithms and techniques. Although TSP is difficult to solve exactly and optimally, heuristic methods such as the Genetic Algorithm can provide adequate and efficient solutions in a short amount of time.

2.3 Genetic Algorithms

Genetic Algorithm is one optimization method based on the principles of natural

selection in biological evolution. This method is used to find the best solution to complex optimization problems by modeling solutions as individuals in a population, and implementing selection, recombination, and mutation processes on that population. The genetic algorithm process starts with creating an initial population of randomly generated individuals. Each individual is defined as a possible solution to the given optimization problem. Next, each individual is evaluated based on certain performance criteria, such as the objective function, which determines the solution's quality (Sutojo et al., 2011).

Then, the selection process takes the individuals with the best performance to be used as parents in the recombination and mutation process. Recombination combines good parts of parents to produce better offspring, while mutation introduces random changes into individuals to search for better solutions and prevent the population from getting trapped in local minima. After the recombination and mutation process, the population is tested again and their performance is evaluated. Individuals with the best performance are selected again as parents for the next generation, and the recombination and mutation process is repeated. This process is repeated until the specified stopping criteria is reached, such as a certain number of generations or a sufficiently good performance (Siahaan, 2016b).

Genetic Algorithm is often used in complex optimization problems, such as the Traveling Salesman Problem and the Vehicle Routing Problem. Although Genetic Algorithm can provide globally optimal solutions, the search process can take a long time and require significant computational resources. Therefore, Genetic Algorithm is often used to find solutions that are close enough to the optimal solution in a reasonable amount of time (Zhao et al., 2016).

METHODOLOGY

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and operations research. The goal of the TSP is to find the shortest possible route that visits a set of cities exactly once and returns to the starting city. One popular approach to solving the TSP is using a Genetic Algorithm (GA), which is a heuristic optimization technique inspired by natural selection and genetics. There are several steps in methodology for solving the Traveling Salesman Problem using Genetic Algorithm such as:

1. Define the problem: The Traveling Salesman Problem is to find the shortest possible route that visits every node exactly once and returns to the starting node.
2. Represent the problem: Represent the problem as a set of nodes, where each node represents a city and the distances between the cities are represented by a distance matrix.
3. Initialize the population: Create an initial population of candidate solutions (i.e., routes) randomly. Each solution consists of a sequence of nodes that represent the order in which the cities are visited.
4. Evaluate the fitness: Evaluate the fitness of each solution by calculating its total distance using the distance matrix.
5. Selection: Select the best-fit solutions (i.e., routes with the shortest distance) to be parents for the next generation using selection techniques like tournament selection or roulette wheel selection.
6. Crossover: Perform crossover (also known as recombination) on the selected parents to create offspring solutions. One common method is to use order crossover, where a random subsequence of one parent is copied into the offspring, and the remaining genes are filled in from the other parent.
7. Mutation: Apply mutation to introduce some diversity into the population. One possible mutation operator is swap mutation, where two cities are randomly selected and swapped in the route.
8. Evaluate the fitness of offspring: Evaluate the fitness of the newly created offspring solutions.
9. Replace the old population with the new population: Use elitism to ensure that the best solutions from the previous population are carried over to the next generation. Replace the rest of the population with the offspring solutions.
10. Termination: Repeat steps 5-9 until a stopping criterion is met, such as reaching a maximum number of generations or the best solution has not improved for a certain number of generations.
11. Output the result: Output the best solution found by the algorithm, which represents the shortest possible route that visits every node exactly once and returns to the starting node.

To define the table route for solving the TSP problem using a Genetic Algorithm (GA) with nodes A, B, C, D, E, it needs to create a distance matrix that represents the distance between each pair of nodes. The distance matrix is used to calculate the fitness of each chromosome in the population and to evaluate the performance of the GA algorithm. The cities are A, B, C, D, and E. The journey starts from A and ends at A as well. The distances between nodes are shown in Figure 1.

P.Issn : 2808-859X
E.Issn : 2809-0853

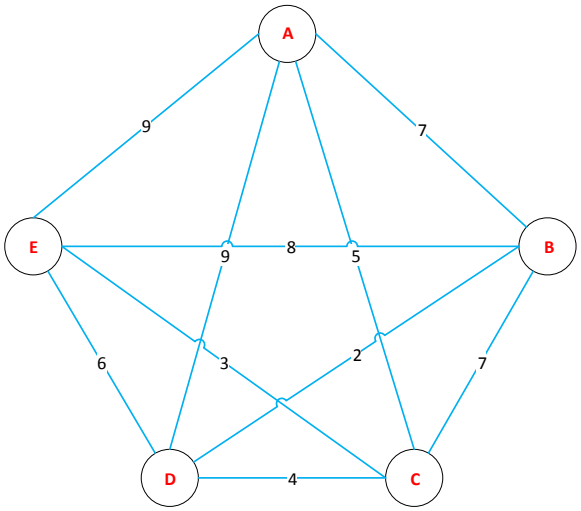

Fig 1. Graph of TSP

Table 1. Distance of each node

	A	B	C	D	E
A	0	7	5	9	9
B	7	0	7	2	8
C	5	7	0	4	3
D	9	2	4	0	6
E	9	8	3	6	0

Table 1 represents the distance between each pair of nodes in a network or graph. In this case, the distance matrix given for nodes A, B, C, D, E shows the distance between each city, which is required to solve the Traveling Salesman Problem.

The values in the distance matrix correspond to the distance between each pair of cities. For example, the value in the row A and column B is 7, which means that the distance between city A and city B is 7 units. Similarly, the value in the row D and column E is 6, which means that the distance between city D and city E is 6 units.

Note that the distance matrix is symmetric, meaning that the distance between city A and city B is the same as the distance between city B and city A. Also, the distance between a city and itself is zero since the distance from a city to itself is always zero.

The distance matrix is important for solving the Traveling Salesman Problem because it is used to calculate the fitness of each chromosome in the population and to

evaluate the performance of the Genetic Algorithm. By using the distance matrix to calculate the fitness of each chromosome, the GA can evolve a population of solutions that are closer to the optimal solution of the TSP problem.

RESULT AND DISCUSSION

A. Initial Chromosomes

In a genetic algorithm for the Traveling Salesman Problem, each solution is represented as a chromosome. A chromosome is a sequence of genes, where each gene represents a city. The order in which the genes appear in the chromosome represents the order in which the salesman visits the cities.

In the given example, the chromosomes represent six different initial solutions to the TSP. Each chromosome has four genes, which means that the salesman must visit four cities in each solution.

The cities represented by the genes in each chromosome are as follows:

1. Chromosome 1: B-D-E-C
2. Chromosome 2: D-B-E-C
3. Chromosome 3: C-B-D-E
4. Chromosome 4: E-B-C-D
5. Chromosome 5: E-C-B-D
6. Chromosome 6: C-D-E-B

These chromosomes represent different possible initial solutions to the TSP, and the genetic algorithm will use them as a starting point to search for an optimal solution. The algorithm will use the principles of genetic variation and selection to evolve and improve these solutions in subsequent generations.

	Chromosomes			
1	B	D	E	C
2	D	B	E	C
3	C	B	D	E
4	E	B	C	D
5	E	C	B	D
6	C	D	E	B

B. Initial Fitness

In a genetic algorithm for the Traveling Salesman Problem, the fitness of a solution refers to its quality or goodness as a solution to the problem. The fitness is typically calculated as the total distance or cost of the solution.

In the given example, it is provided with six initial solutions (chromosomes) to the TSP, and the fitness of each solution is calculated by summing up the distances between the cities in the order specified by the chromosome. The table shows the initial solutions (specified by the routes between the cities) and their corresponding fitness values:

1. Solution 1: AB-BD-DE-EC-CA (7+2+6+3+5 = 23)
2. Solution 2: AD-DB-BE-EC-CA (9+2+8+3+5 = 27)
3. Solution 3: AC-CB-BD-DE-EA (5+7+2+6+9 = 29)
4. Solution 4: AE-EB-BC-CD-DA (9+8+7+4+9 = 37)
5. Solution 5: AE-EC-CB-BD-DA (9+3+7+2+9 = 30)
6. Solution 6: AC-CD-DE-EB-BA (5+4+6+8+7 = 30)

These fitness values represent how good each solution is as a solution to the TSP. The goal of the genetic algorithm is to find a solution with the lowest possible fitness value, which corresponds to the shortest possible distance between the cities. The algorithm achieves this by using the principles of genetic variation and selection to evolve and improve the solutions in subsequent generations.

	Route					**Fitness**
1	AB	BD	DE	EC	CA	
	7	2	6	3	5	23
2	AD	DB	BE	EC	CA	
	9	2	8	3	5	27
3	AC	CB	BD	DE	EA	
	5	7	2	6	9	29
4	AE	EB	BC	CD	DA	
	9	8	7	4	9	37
5	AE	EC	CB	BD	DA	
	9	3	7	2	9	30
6	AC	CD	DE	EB	BA	
	5	4	6	8	7	30

C. Selection

Roulette wheel selection is a popular method for selecting individuals in a genetic algorithm based on their fitness values. In this method, a roulette wheel is created, with each chromosome occupying a section of the wheel that is proportional to its fitness value. The sections are then selected randomly, and the chromosomes that occupy the selected sections are chosen for the next generation.

To apply roulette wheel selection in this example, first calculate the total fitness value of all chromosomes, which is 0.208692 in this case. Then, calculate the proportional fitness value of each chromosome by dividing its fitness value by the total fitness value. The proportional fitness value of each chromosome represents the probability of that chromosome being selected.

Next, create a roulette wheel by dividing the range [0, 1) into sections proportional to the proportional fitness values of the chromosomes. For example, if the proportional fitness value of chromosome 1 is 0.043478, assign it a section of the wheel that spans from 0 to 0.043478 on the wheel.

Finally, select chromosomes by spinning the roulette wheel and selecting the chromosome that occupies the section of the wheel where the wheel stops. This process is repeated until it has selected the desired number of chromosomes for the next generation. In this example, the proportional fitness values and the sections on the roulette wheel for the chromosomes are:

$$0.043478\ (0.000000 - 0.043478)$$
$$0.037037\ (0.043478 - 0.080515)$$
$$0.034483\ (0.080515 - 0.114998)$$
$$0.027027\ (0.114998 - 0.142025)$$
$$0.033333\ (0.142025 - 0.175358)$$
$$0.033333\ (0.175358 - 0.208692)$$

	Chromosome		
1	1	23	0.043478
2	1	27	0.037037
3	1	29	0.034483
4	1	37	0.027027
5	1	30	0.033333
6	1	30	0.033333
Total			**0.208692**

D. Probability

In the context of genetic algorithms, probability refers to the likelihood of a chromosome being selected for reproduction and possibly mutation in the next generation. In the roulette wheel selection process, the probability of selecting a particular chromosome is proportional to its fitness value relative to the total fitness value of all chromosomes.

In the table you provided, the first column shows the probabilities calculated for each chromosome using the formula: probability = fitness value / total fitness value. The second column shows the total fitness value of all chromosomes, which is simply the sum of the fitness values. The third column shows the cumulative probabilities, which are obtained by summing up the probabilities of all chromosomes up to a certain point. These cumulative probabilities will be used to randomly select chromosomes for reproduction in the next generation.

For example, if a random number between 0 and 1 is generated and falls between 0 and 0.208337, chromosome 1 will be selected for reproduction. Similarly, if the random number falls between 0.208337 and 0.385809 (which is the sum of the probabilities of chromosome 1 and chromosome 2), then chromosome 2 will be selected, and so on.

	Probabilty		
1	0.043478	0.208692	0.208337
2	0.037037	0.208692	0.177472
3	0.034483	0.208692	0.165233
4	0.027027	0.208692	0.129507
5	0.033333	0.208692	0.159725
6	0.033333	0.208692	0.159725

E. Cumulative Probability

Cumulative probability refers to the running total of probabilities at each selection point during roulette wheel selection in genetic algorithm. In the table provided, the cumulative probabilities for each chromosome have been calculated using the probabilities calculated in the previous table.

For example, for the first chromosome, the cumulative probability is 0.208337 which is equal to its probability. For the second chromosome, the cumulative probability is calculated by adding the probability of the second chromosome to the cumulative

probability of the first chromosome (0.177472 + 0.208337 = 0.38581). The cumulative probability for the last chromosome is always 1 as it includes all the previous probabilities. These cumulative probabilities are used to determine which chromosome will be selected for the next generation during the selection process in the genetic algorithm.

	Cumulative		
1	0.208337	0	0.208337
2	0.208337	0.177472	0.38581
3	0.38581	0.165233	0.551043
4	0.551043	0.129507	0.68055
5	0.68055	0.159725	0.840275
6	0.840275	0.159725	1

F. Crossover

In genetic algorithms, crossover is a genetic operator used to combine genetic information from two parents to produce offspring. The crossover operator randomly selects a point along the chromosome, known as the crossover point, and exchanges the genetic material of the parents to generate new offspring.

In the given table, the random numbers generated for crossover indicate the probability of performing the crossover operation. For example, for chromosome 1, the random number generated is 0.452, which is greater than the crossover probability set by the algorithm. Therefore, no crossover will be performed on chromosome 1.

On the other hand, for chromosome 2, the random number generated is 0.209, which is less than the crossover probability. Therefore, crossover will be performed on chromosome 2. The same process is applied to all the chromosomes in the population.. The crossover probability value is 0.25.

	Random
1	0.452
2	0.209
3	0.221
4	0.875
5	0.770
6	0.133

Crossover is a genetic operator used in genetic algorithms to combine genetic information from two parent chromosomes to create new offspring chromosomes. In the context of the given TSP problem, crossover is used to combine the genetic information from two selected parent chromosomes to create new offspring chromosomes that can potentially lead to better solutions.

The crossover points are determined by generating random numbers, usually between 0 and 1, to determine where the crossover will occur. In the given example, the random numbers generated for each pair of parent chromosomes are:

1. 0.452
2. 0.209
3. 0.221
4. 0.875
5. 0.770
6. 0.133

These random numbers are used to determine the crossover points for each pair of parent chromosomes. For example, for parent chromosomes 1 and 2, the crossover point is determined by the random number 0.452. This means that the first 45.2% of the genetic information from parent 1 will be retained, and the remaining 54.8% of the genetic information will come from parent 2.

After the crossover points have been determined, the genetic information from the parent chromosomes is combined to create the new offspring chromosomes. The resulting offspring chromosomes are then added to the population for the next generation of the genetic algorithm.

These new offspring chromosomes will be used in the next generation of the genetic algorithm to potentially find even better solutions to the TSP problem.

Old	New	Chromosomes			
1	2	D	B	E	C
2	1	B	D	E	C
3	3	C	B	D	E
4	6	C	D	E	B
5	5	E	C	B	D
6	4	E	B	C	D

G. Mutation

Mutation is a genetic operator that introduces random changes in an individual chromosome to explore new search spaces. In the context of the genetic algorithm, mutation is applied to a selected chromosome with a certain probability to maintain diversity in the population.

In the given example, mutation has been applied to the chromosomes with a probability of 0.2. The result of the mutation is a new set of chromosomes that are slightly different from the parent chromosomes. The mutation operator randomly swaps two adjacent genes in the chromosome.

In the resulting chromosomes after the mutation, the order of the nodes has changed from their original position. For example, in chromosome 1, the genes BD have been swapped to give the new chromosome DB. Similarly, in chromosome 4, the genes BC have been swapped to give the new chromosome CB.

Overall, the effect of mutation is to introduce diversity in the population, which can help the algorithm to escape from local optima and explore new regions of the search space.. The mutation rate is 0.2.

	Chromosomes			
1	D	B	E	C
2	B	D	E	C
3	C	B	D	E
4	C	D	E	B
5	E	C	B	D
6	E	B	C	D

H. Fitnes Value

During the initial generation, it was observed that the minimum fitness value remained constant. It is assumed that this minimum value will persist throughout subsequent generations up to the Nth generation. Despite the fact that the calculation was only carried out until the first generation, the genetic algorithm process yielded a solution that was close to optimal. The final result of this process identified the optimal route to be A, B, D, E, C, A, which has the shortest distance.

	Route					Fitness
1	AD	DB	BC	CE	EA	
	9	2	7	3	9	30
2	AB	BD	DE	EC	CA	
	7	2	6	3	5	23
3	AC	CE	ED	DB	BA	
	5	3	6	2	7	23
4	AE	EC	CB	BD	DA	
	9	3	7	2	9	30
5	AD	DB	BC	CE	EA	
	9	2	7	3	9	30
6	AE	ED	DB	BC	CA	
	9	6	2	7	5	29

From the table, it appears that the shortest distance is achieved by Route 2, which follows the path A-B-D-E-C-A and has a fitness value of 23. This route involves traveling from A to B, then to D, followed by E, then to C, and back to A. The fitness value of 23 likely represents the total distance or time required to travel this path.

Compared to the other routes, Route 2 has the lowest fitness value, indicating that it is the most efficient path in terms of the performance metric being used. For example, if the fitness value represents distance, then Route 2 would be the shortest path among all the routes. Similarly, if the fitness value represents time, then Route 2 would be the fastest route.

It is important to note that the optimal path or solution depends on the problem being addressed and the specific performance metric being used. Different optimization problems may have different criteria for what constitutes an optimal solution. Therefore, it is important to understand the context of the problem and the metric being used to evaluate performance when selecting a solution.

CONCLUSION

Overall, the results suggest that genetic algorithms can be effective tools for solving optimization problems, particularly those that involve searching through a large space of potential solutions. However, it is important to note that the quality of the solution

obtained may depend on various factors, such as the specific problem being addressed, the performance metric being used, and the parameters of the genetic algorithm (e.g., population size, mutation rate). Therefore, it may be necessary to fine-tune the algorithm and explore different variations in order to obtain the best possible solution.

Based on the experimental results provided in the table, it appears that the genetic algorithm was able to identify a near-optimal solution for the problem of finding the shortest route among a set of destinations. The algorithm was able to generate several potential routes (Routes 1-6) and evaluate their fitness based on some performance metric (e.g., distance, time).

Among the routes generated, Route 2 had the lowest fitness value, indicating that it was the most efficient path. This route involved traveling from A to B, then to D, followed by E, then to C, and back to A. This result demonstrates that genetic algorithms can be a powerful tool for solving optimization problems like the traveling salesman problem, where the objective is to find the shortest possible route that visits a set of destinations.

The quality of the solution obtained may depend on various factors, such as the specific problem being addressed, the performance metric being used, and the parameters of the genetic algorithm (e.g., population size, mutation rate). Therefore, it may be necessary to fine-tune the algorithm and explore different variations in order to obtain the best possible solution for a particular problem.

REFERENCES

Borna, K., & Hashemi, V. H. (2014). An Improved Genetic Algorithm with a Local Optimization Strategy and an Extra Mutation Level for Solving Traveling Salesman Problem. *International Journal of Computer Science, Engineering and Information Technology*, *4*(4), 47–53. https://doi.org/10.5121/ijcseit.2014.4405

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

El Hassani, H., Benkachcha, S., & Benhra, J. (2015). New Genetic Operator (Jump Crossover) for the Traveling Salesman Problem. *International Journal of Applied Metaheuristic Computing*, *6*(2), 33–44. https://doi.org/10.4018/IJAMC.2015040103

Gabriel, J. (2016). *Artificial Intelligence: Artificial Intelligence for Humans*. CreateSpace Independent Publishing.

Siahaan, A. P. U. (2016a). Genetic Algorithm in Hill Cipher Encryption. *American*

International Journal of Research in Science, Technology, Engineering & Mathematics, *15*(1), 84–89.

Siahaan, A. P. U. (2016b). Scheduling Courses using Genetic Algorithms. *International Journal of Computer Applications*, *153*(3), 20–25.

Siahaan, A. P. U., Rusiadi, Kan, P. L. E., Azir, K. N. F. K., & Amir, A. (2018). Prim and Genetic Algorithms Performance in Determining Optimum Route on Graph. *International Journal of Control and Automation*, *11*(6), 109–122. https://doi.org/10.14257/ijca.2018.11.6.11

Sutojo, T., Mulyanto, E., & Suhartono, V. (2011). *Kecerdasan Buatan*. Andi Offset.

Tabatabaee, H. (2015). Solving the Traveling Salesman Problem using Genetic Algorithms with the New Evaluation Function. *Bulletin of Environment, Pharmacology and Life Sciences*, *4*(11), 124–131. https://pdfs.semanticscholar.org/1fdc/eab75bd110c594e9af2afbf399f901b04471.pdf

Zhao, J., Shi, R., Sai, L., Huang, X., & Su, Y. (2016). Comprehensive genetic algorithm for ab initio global optimisation of clusters. *Molecular Simulation*, *42*(10), 809–819. https://doi.org/10.1080/08927022.2015.1121386

Zhou, A.-H., Zhu, L.-P., Hu, B., Deng, S., Song, Y., Qiu, H., & Pan, S. (2018). Traveling-Salesman-Problem Algorithm Based on Simulated Annealing and Gene-Expression Programming. *Information*, *10*(1), 7. https://doi.org/10.3390/info10010007