Genetic Algorithm Application in Route Optimization Using Google Maps API

Ivan Hintoro[*1], Sandy Kosasi[2], David[3], Susanti Margaretha Kuway[4], Gat[5]
[1,2,3,4,5]STMIK Pontianak, Pontianak, Indonesia
E-mail: [*1]**ivanhintoro@gmail.com**, [2]sandykosasi@stmikpontianak.ac.id,
[3]david@stmikpontianak.ac.id, [4]shantykuway@stmikpontianak.ac.id,
[5]gat@stmikpontianak.ac.id

Abstract

Route planning from several locations is a common problem encountered. Google Maps provides a feature to plan routes from multiple location but does not provide a feature to find the optimal route from these locations. Based on the survey, 61.9% of respondents have a low level of confidence when planning the optimal route from several locations. So we need software that can help in planning the route. This research utilizes genetic algorithms to help plan routes from several locations. The genetic algorithm used includes components in the form of chromosomes, fitness values, selection, crossover, and mutations. In addition, Google Maps API used as the data source that provides maps, directions, and distance matrices. The design method used extreme programming. The software is made using HTML, CSS, PHP, and JavaScript programming languages with MySQL DBMS. In addition, Black-box and White-box testing method is used in this research. The results of this study are route optimization software that works by using genetic algorithm components to obtain optimal route from several locations with the advantages of using Google Maps API and selecting location points and starting points that are flexible. This study aims to highlight the benefits, limitations, and future prospects of route optimization. The findings of this research will contribute to assist decision-makers in adopting effective strategies related to route optimization.

Keywords *— Genetic Algorithm, Optimization, Routes*

1. INTRODUCTION

The rapid development of technology accompanied by advances in science has led to the emergence of many tools and knowledge that can help people's daily lives. One of them is in the field of information technology, especially artificial intelligence. The development of artificial intelligence technology is being discussed by technology experts. Artificial intelligence is called the driver of the industrial revolution 4.0 which is now starting to enter the era of Digital Disruption 4.0 and can create new faces in the development of machines that have been integrated by the internet[1].

In the field of artificial intelligence, a method for solving optimization problems is developed, namely genetic algorithms. Genetic Algorithms are a class of general purpose (domain independent) search methods which strike a remarkable balance between exploration and exploitation of the search space[2]. GA was inspired by population genetics (heredity and gene frequencies), and evolution at the population level besides the Mendelian understanding of the structure (chromosomes, genes, alleles) and process/mechanisms (recombination and

mutation)[3]. Genetic algorithms have been through many applications in their application to get good results in several studies[4]. Genetic algorithms are used to find solutions to optimization problems that generally cannot be solved efficiently by conventional algorithms. The cycle of the genetic algorithm based on this concept has stages starting with initializing a solution at random, then recombining two or more solutions with a recombination operator, then selecting the best solution for the next generation. The iteration of this stage is then continued until the termination condition is met[5].

One of the internet technologies that is often used is the World Wide Web or more commonly referred to as the web. Web applications are known as applications that can be accessed via a web browser and through networks such as the internet or intranet, and the ability to update and maintain web applications without having to distribute and install software on client computers is an advantage of this technology[6]. Web-based software is developing very rapidly because it has advantages in the form of easy access from anywhere without installation, the use of servers for data storage, cross-platform nature, lightweight specification requirements from the client side, and good security because the software runs on a browser[7].

Currently, the use of maps has been digitized to make them easier to use. Therefore, Google Maps was launched as a tool for directions and places that can cover almost all regions in the world[8]. The Google Maps API allows developers to build applications by leveraging Google Maps features across multiple platforms. The API provided by Google Maps gives developers access to services in the form of displaying maps and getting data directly from the user when inputting a location and retrieving route distance matrix data from several points of origin and destination.

One of the problems that exist in artificial intelligence is the optimization problem which refers to computing to maximize or minimize the objective function by considering the problem[9]. Route optimization is one type of optimization problem that often arises in everyday life. This problem can be described as finding the shortest route for someone who visits several places exactly once. Although by using conventional algorithms, the best solution can be found, this method is not effective because the calculation process to explore all existing solutions will take a very long time, especially if the problem being solved is more complex[10]. The more points that must be passed, the more inefficient the use of conventional algorithms is because of the exponential complexity of the algorithm.

The main problem of this research is the difficulty faced by someone when planning a route from many locations to be visited. Planning an efficient route with a large number of location points will be difficult for someone to do, therefore we need a tool in the form of software that can help solve this problem. Optimization of the sequence of visits to location points will then become the main problem that will be solved by the research conducted.

The research conducted by Utami[11], Santoso[12], and Prakasa[13] has demonstrated the effectiveness of utilizing genetic algorithms for solving route optimization problems. Building upon these earlier studies, the present research extends the existing knowledge by leveraging the capabilities of the Google Maps API to acquire accurate location data. Moreover, this

study introduces a novel element of flexibility in selecting the starting point, which was previously constrained in the prior research endeavors. By incorporating the Google Maps API, this research enhances the precision and reliability of the route optimization process, enabling more accurate and adaptable solutions. The utilization of a more flexible starting point selection adds a practical dimension to the research, as it accommodates real-world scenarios where the starting location may vary. Overall, this research represents a valuable progression in the field of route optimization, combining the power of genetic algorithms with the convenience and accuracy of the Google Maps API to deliver improved results.

2. RESEARCH METHOD

The form of research used is a case study where the case discussed will be studied in depth. The research method used is an experimental research method used to find the effect of certain treatments on others under controlled conditions[14].

2.1. Data Collection

Data collection methods are divided into primary and secondary data collection. The primary data in this study was obtained from distributing online questionnaires using Google Form to respondents so that data collection is more practical and time efficient. This primary data is in the form of the respondent's level of confidence in route planning. Secondary data in this study was obtained by conducting a documentation study in the form of collecting information and references from libraries, articles in research journals, and documents relevant to the research. Aside from those, Google Maps API is also used as data source for the software.

2.2. Software Design

The software design method used is the Extreme Programming (XP) method. Extreme Programming implies a set of rules and practices that occur in the context of four framework activities: planning, design, coding, and testing[15].

Software design in this study begins with identifying software requirements. The main requirement of the software to be designed is that the software must be able to find the optimal route from a set of location points using the Google Maps API and genetic algorithms. Apart from these main requirements, the software also has other needs in the form of supporting features such as login systems, location management, and route management. Therefore, the software must also be able to validate user credentials, provide management features in the form of adding, deleting, and changing location and route information.

The next step after identifying the software requirements is the genetic algorithm design stage. Genetic Algorithm is a search procedure based on the process of natural selection and genetics. It is gradually being used in applications in difficult search, optimization, and machine learning problems, across a wide spectrum of human endeavors[16]. The design of this route optimization software has components of the genetic algorithm used. These constituent components are described in the flow chart model in Figure 1 below.

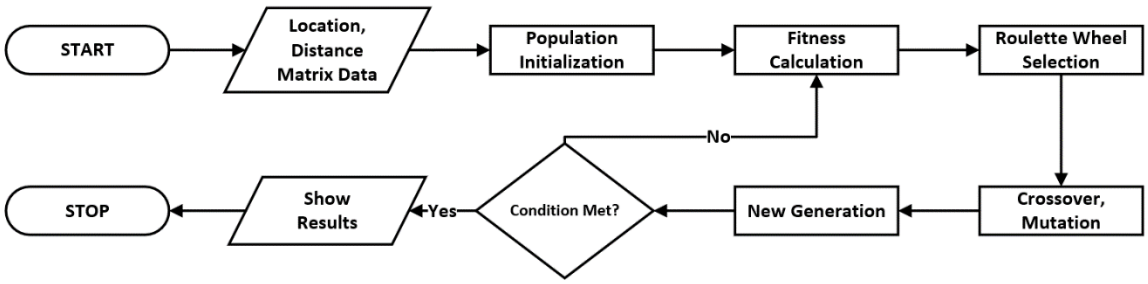


Figure 1. Genetic Algorithm Model

In a simple genetic algorithm, many individuals in a population are maintained. Each individual has two properties: chromosomes and fitness. After getting the fitness of all individuals, the selection process is used to generate mating ponds. Individuals with higher qualities should have a higher chance of being selected into the mating pond so that the good ones will have more opportunities to breed. In general, two parents may be randomly selected from the mating pool to produce offspring and each offspring may undergo minor changes to become a new individual. Then the newly generated population replaces the old one and another generation begins[17].

Genetic algorithms solve problems on the basis of optimizing solutions. A solution is symbolized as a chromosome. Generally, these chromosomes are the result of processing a solution consisting of an ordered data set. This software is designed to use a sequence of location data as a solution, where these locations have a distance from one point to another. Each distance between these points is then compiled into a distance matrix so that it can be processed by genetic algorithms.

This distance matrix is obtained using the Google Maps API, or more specifically the Distance Matrix API provided by the Google Maps API service. There are several open source and proprietary alternatives to the Google Maps JavaScript API, but what makes this API special for developers is that it has a complete solution with base maps, overlays, and technical capabilities[18]. By using this API, the system can make requests for the distance matrix between locations from the inputted location points. The inputted location points are a collection of latitude and longitude coordinate. The distance matrix obtained is in the form of data in JSON format which has distance and time, each of which has units, namely meters and seconds.

Chromosomes in this study represent the solution in the form of a sequence of points visited. Permutation Encoding is used to encode each location point so that it has an index value in the form of an integer which will then be compiled to represent the order of visiting points as a solution. The length of the chromosome depends on the number of location points entered by the user with a maximum of 10 points because it is a limitation of the API used.

Chromosomes also have some constraints that need to be considered. These constraints include chromosomes containing location points that are inputted and represented by integer numbers, each location only exists on the chromosome exactly once, the first location on the

chromosome is the starting point entered by the user and during the optimization process will not change, the length of the chromosome must match the number of location points entered.

The chromosome decoding process in this study was carried out by returning the solution by converting the integer number back into a location point so that a sequence of location points was obtained as a solution.

Each individual has a value that represents how good a solution is, namely the fitness value. To get the fitness value, we need the fitness function that measures the quality of the solution that has been generated by the genetic algorithm[19]. The fitness function design is part of the modeling process of the overall optimization approach. The fitness value in this route optimization is related to the route distance. Because the purpose of using genetic algorithms in this study is to minimize distance, the smaller the distance value in a solution, the better the solution. This distance value is obtained by adding up all the distances between points obtained from the distance matrix for an individual.

The fitness value is then obtained by equation 1 as follows.

$$Fitness = \frac{1}{TotalDistance+1} \quad (1)$$

Then normalization is done on the fitness value using the following equation 2.

$$Fitness = \frac{Fitness}{TotalFitness} \quad (2)$$

Selection is the process of selecting individuals who will become parents of the next generation[20]. The selection method used in this genetic algorithm is roulette wheel selection. The roulette wheel selects individuals based on the probability of the individual's fitness value.

The crossover in this study uses a modification of the cut-point crossover method for the Permutation Encoded Chromosome. This crossover model is done by selecting a cut point at random on one parent chromosome, then inserting genes from other parent chromosomes in order, provided that only genes that are not found on the first parent chromosome are included. Here in Figure 2 the crossover model used.

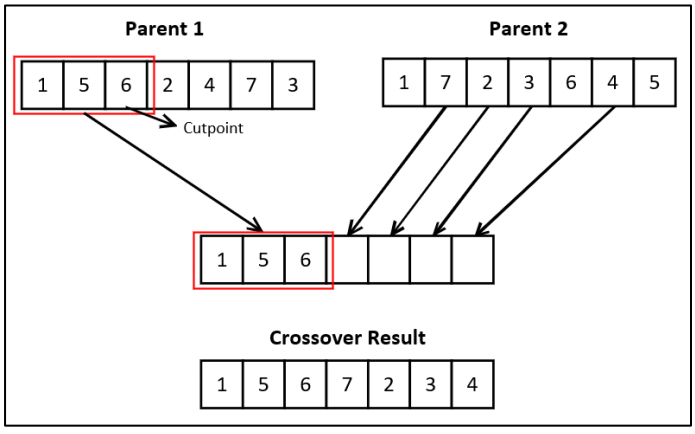


Figure 2. Crossover Model

Mutation is one way to get new individuals apart from crossover. Mutations are intended to control the level of diversity of the population so that the population does not converge quickly and is trapped at the local optimum. Every individual has the possibility to experience a mutation, the magnitude of this possibility is influenced by a factor, namely the mutation rate. If an individual gets a random value within the range of the mutation rate, then that individual will carry out the mutation process.

Mutations in this genetic algorithm have an exchange point mutation model. Exchange point mutation has a process, namely randomly selecting two points on a chromosome and then the values at both points will be exchanged. The following in Figure 3 is a modeling of the exchange point mutation used.

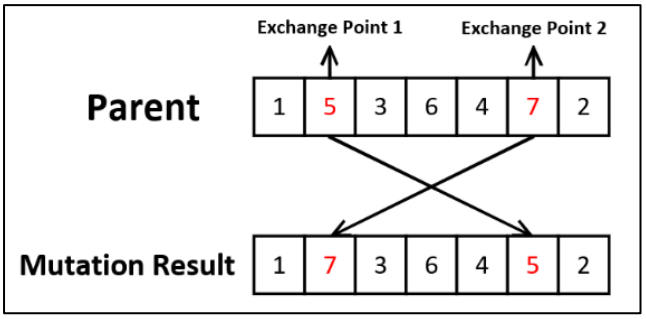


Figure 3. Mutation Model

After designing the genetic algorithm, then software modeling is carried out. The software is modeled using the software architecture shown in Figure 4 below.

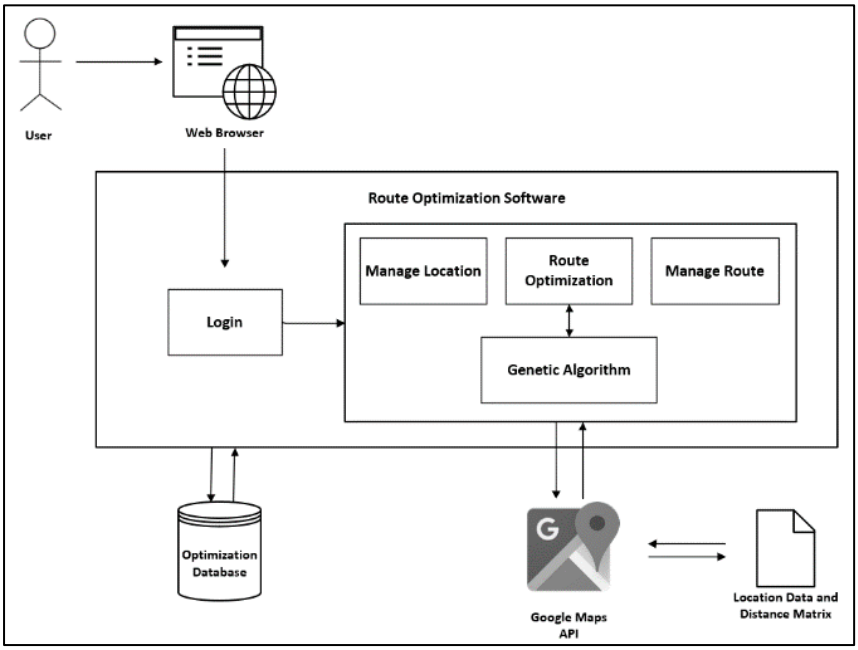


Figure 4. Software Architecture

As shown in Figure 4, the software is accessed by the user through a web browser. The user first logs in before being able to perform location management, route optimization, and

route management. For the overall use of the feature, access to the optimization database will be carried out, then specifically for the location management feature, route optimization, and route management will access the data provided by the Google Maps API in the form of location data and distance matrix.

Class diagrams are used to describe the structure of a system and also show the static structure of classifiers in a system. The following in Figure 5 is a class diagram of the software to be designed.

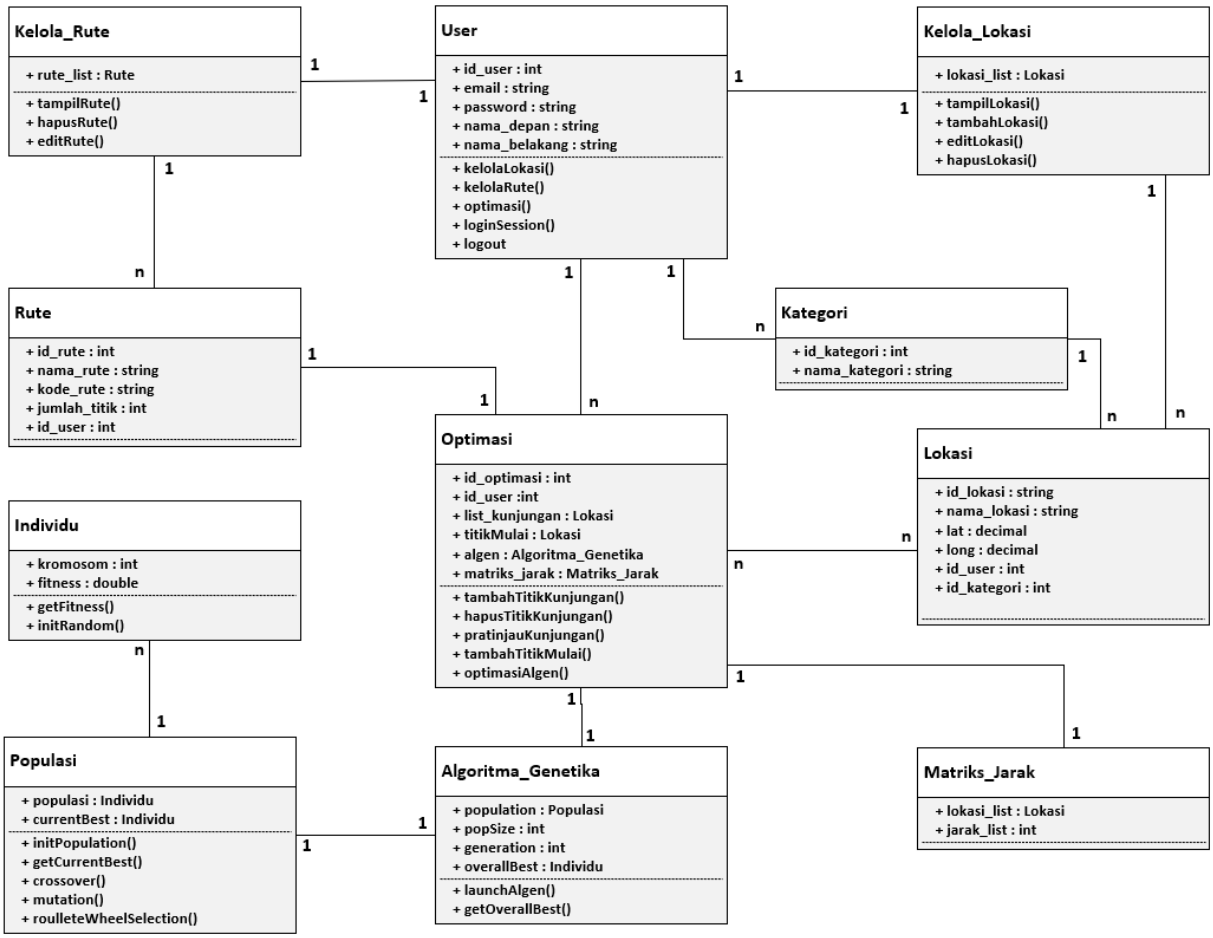


Figure 5. Class Diagram

3. RESEARCH RESULTS AND DISCUSSION

Through the research, it was found that genetic algorithms offered significant advantages over traditional optimization methods when applied to route planning. By using a population of potential solutions and iteratively applying genetic operators such as selection, crossover, and mutation, the algorithm was able to generate optimal or near-optimal routes efficiently. Based on the software design that has been modeled, then the interface of the software is designed. The following in Figure 6 is the result of the interface of the designed software.

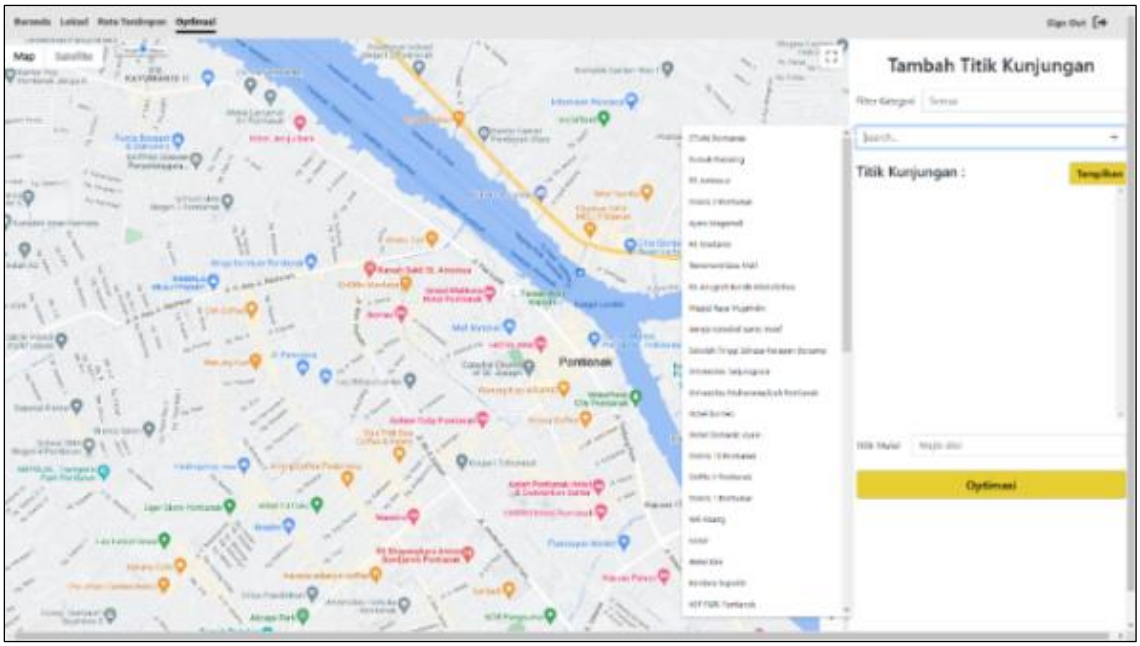
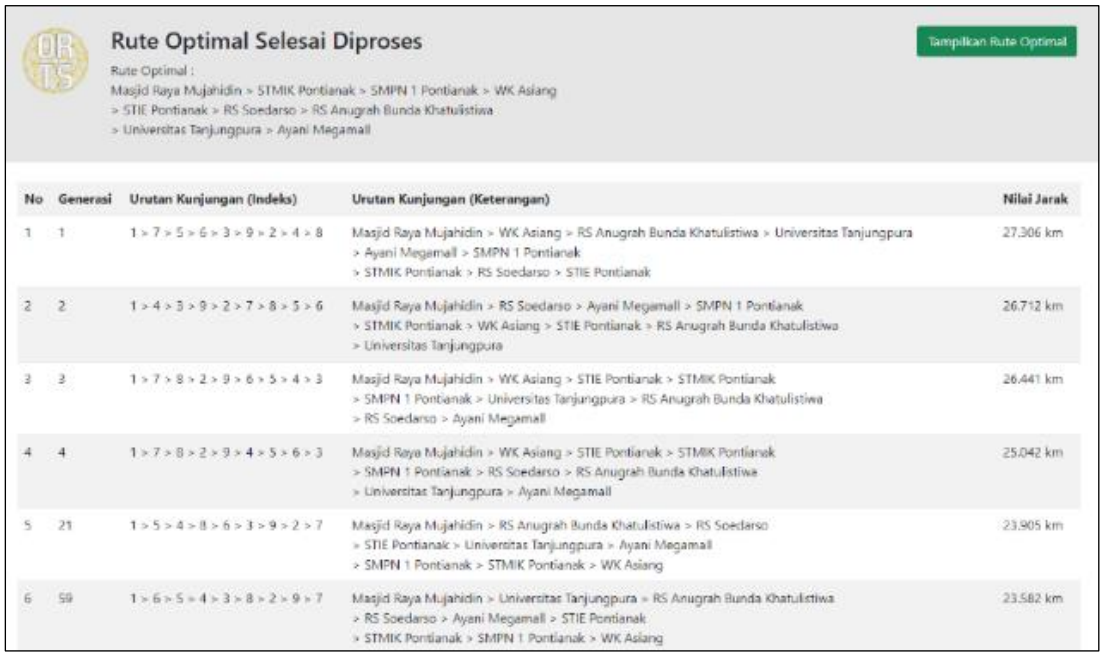


Figure 6. Route Optimization Page

Users perform optimization by adding visiting points. When the user adds points of visit, then the page will show the points of visits to the list. After the user adds the location points and adds the starting point of the route, then the user can click on optimization to start the optimization process. The optimization process is carried out using a genetic algorithm to find the optimal solution from the results of random solutions applied by crossover and mutation operators. Each generation will produce a population of solutions that can have a better fitness value than the previous generation. Each individual who has a better score than the individual of the previous generation will be displayed on the optimization processing page shown in Figure 7 below.



No	Generasi	Urutan Kunjungan (Indeks)	Urutan Kunjungan (Keterangan)	Nilai Jarak
1	1	1 > 7 > 5 > 6 > 3 > 9 > 2 > 4 > 8	Masjid Raya Mujahidin > WK Asiang > RS Anugrah Bunda Khatulistiwa > Universitas Tanjungpura > Ayani Megamall > SMPN 1 Pontianak > STMIK Pontianak > RS Soedarso > STIE Pontianak	27.306 km
2	2	1 > 4 > 3 > 9 > 2 > 7 > 8 > 5 > 6	Masjid Raya Mujahidin > RS Soedarso > Ayani Megamall > SMPN 1 Pontianak > STMIK Pontianak > WK Asiang > STIE Pontianak > RS Anugrah Bunda Khatulistiwa > Universitas Tanjungpura	26.712 km
3	3	1 > 7 > 8 > 2 > 9 > 6 > 5 > 4 > 3	Masjid Raya Mujahidin > WK Asiang > STIE Pontianak > STMIK Pontianak > SMPN 1 Pontianak > Universitas Tanjungpura > RS Anugrah Bunda Khatulistiwa > RS Soedarso > Ayani Megamall	26.441 km
4	4	1 > 7 > 8 > 2 > 9 > 4 > 5 > 6 > 3	Masjid Raya Mujahidin > WK Asiang > STIE Pontianak > STMIK Pontianak > SMPN 1 Pontianak > RS Soedarso > RS Anugrah Bunda Khatulistiwa > Universitas Tanjungpura > Ayani Megamall	25.042 km
5	21	1 > 5 > 4 > 8 > 6 > 3 > 9 > 2 > 7	Masjid Raya Mujahidin > RS Anugrah Bunda Khatulistiwa > RS Soedarso > STIE Pontianak > Universitas Tanjungpura > Ayani Megamall > SMPN 1 Pontianak > STMIK Pontianak > WK Asiang	23.905 km
6	59	1 > 6 > 5 > 4 > 3 > 8 > 2 > 9 > 7	Masjid Raya Mujahidin > Universitas Tanjungpura > RS Anugrah Bunda Khatulistiwa > RS Soedarso > Ayani Megamall > STIE Pontianak > STMIK Pontianak > SMPN 1 Pontianak > WK Asiang	23.582 km

Figure 7. Optimization Processing Page

On this page, the solution for the calculation of the genetic algorithm is displayed. The results of the research demonstrated that the genetic algorithm approach achieved notable improvements in route optimization, particularly in scenarios involving multiple waypoints or destinations. The algorithm effectively minimized travel distances and reduced the overall time required for reaching all the destinations.

The best solution for a generation will be displayed with a record that only the generation that has the best solution changes will be displayed. In addition, the distance value from the best solution in that generation is also displayed. Users can continue the process by clicking on the 'show optimal route' button so that the best route obtained by the genetic algorithm will be displayed on the optimization results page as shown in Figure 8.

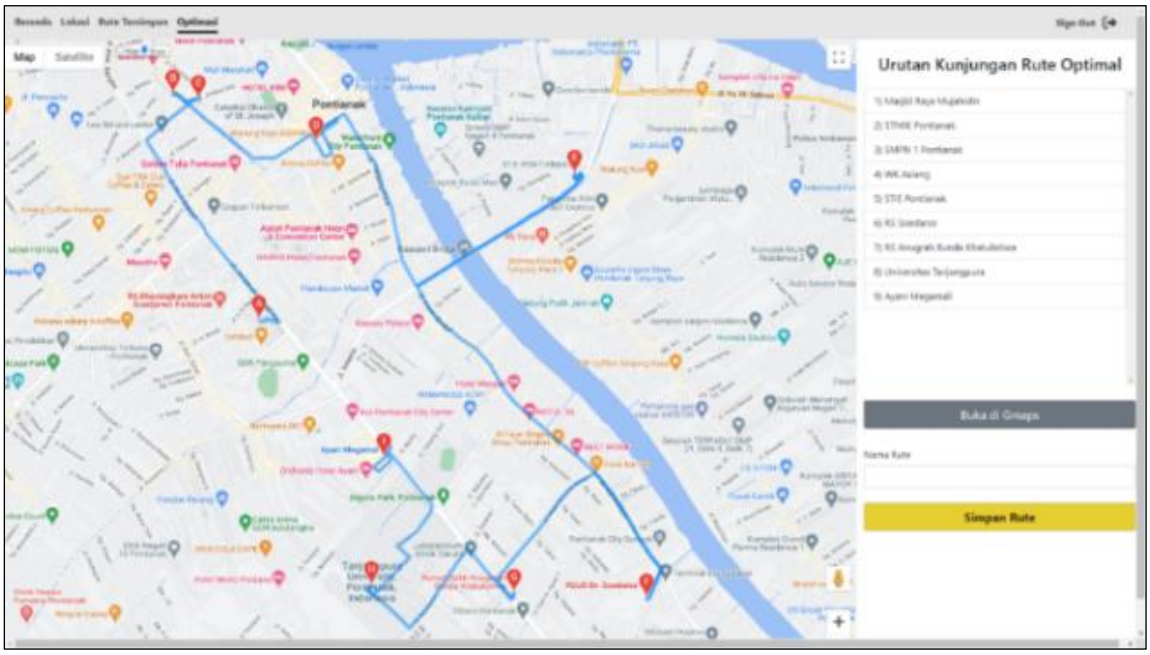


Figure 8. Optimal Route Results Page

Case testing is done to see the performance of the software against an example problem. In this case testing, optimization of 8 location points was carried out with location data as shown in Table 1.

Table 1. Location Data

#	Location Name	Latitude	Longitude
1	Rumah Radakng	-0,02580	109,33060
2	RS Antonius	-0,02026	109,32677
3	Pasar Flamboyan	-0,04038	109,3452
4	Ayani Megamall	-0,05193	109,34587
5	RS Yarsi	-0,03654	109,35722
6	Gereja Katedral Santo Yosef	-0,02741	109,3383
7	Universitas Tanjungpura	-0,06074	109,34504
8	Hotel Orchardz Ayani	-0,05331	109,34445

Based on the location latitude and longitude value, by making a request to the API, the distance between each point is obtained. it is this collection of distance values that is entered

into the distance matrix. Based on the data used in this case testing, the distance matrix is shown in Table 2.

Table 2. Distance Matrix

	1	2	3	4	5	6	7	8
1	0	4006	4793	4402	5948	3378	5996	4367
2	4175	0	3924	6746	5282	2697	6783	6644
3	4218	4553	0	3166	1770	1796	3203	3064
4	4359	4451	2810	0	3965	3775	4013	441
5	5824	5388	1759	4666	0	3563	4704	4565
6	3319	2283	1806	5267	4029	0	5304	5165
7	5954	6046	4404	2031	4736	5369	0	1929
8	4244	5092	3451	921	4606	4415	3977	0

If there is a route solution with the order of visits as shown in Figure 9 as follows.

Pasar Flamboyan → Ayani Megamall → Universitas Tanjungpura → Rumah Radakng → RS Antonius → RS Yarsi → Hotel Orchardz Ayani → Gereja Katedral Santo Yosef

Figure 9. Solution Visit Order

Then the chromosome encoding of the solution can be carried out as shown in Figure 10 below.

Visit Order	1st Point	2nd Point	3rd Point	4th Point	5th Point	6th Point	7th Point	8th Point
	Pasar Flamboyan	Ayani Megamall	Universitas Tanjungpura	Rumah Radakng	RS Antonius	RS Yarsi	Hotel Orchardz	Gereja Katedral
Gene	3	4	7	1	2	5	8	6
Chromosome	[34712586]							

Figure 10. Chromosome Encoding

Then the initial generation population is generated randomly. The following is the population generated in Table 3.

Table 3. Initial Population

#	Chromosome
1	[3 4 7 1 2 5 8 6]
2	[3 8 4 2 5 1 6 7]
4	[3 7 4 5 6 2 1 8]
5	[3 7 8 2 1 5 4 6]
6	[3 6 2 7 1 5 4 8]
7	[3 2 5 1 8 6 7 4]
8	[3 8 7 1 4 6 2 5]
9	[3 4 1 5 8 7 2 6]
10	[3 2 1 7 5 4 8 6]

A fitness calculation is performed on each individual in the population. Fitness is calculated by calculating the distance in advance by adjusting the route of the visit to the distance matrix. The following in Figure 11 is the process of calculating the distance to a solution.

Individual	1							
Chromosome	[3 4 7 1 2 5 8 6]							
Gene	3	4	7	1	2	5	8	6
Route	3 → 4	4 → 7	7 → 1	1 → 2	2 → 5	5 → 8	8 → 6	
Points Distance	3166	4013	5954	4006	5282	4565	4415	
Total Distance	31401							

Figure 11. Distance Calculation

Then the fitness calculation on the individual is carried out so that the fitness value of individual 1 with chromosome [3 4 7 1 2 5 8 6] which has a distance of 31,401 m is obtained as follows.

$$Fitness = \frac{1}{TotalDistance+1} = \frac{1}{31401+1} = 0{,}0000318451 \quad (3)$$

A fitness calculation is performed for each individual in the initial population. The following Table 4 is the fitness value of each individual in the initial population.

Table 4. Initial Population Fitness

#	Chromosome	Total Distance	Fitness
1	[3 4 7 1 2 5 8 6]	31401	0,0000318451
2	[3 8 4 2 5 1 6 7]	28224	0,0000354296
3	[3 7 4 5 6 2 1 8]	23587	0,0000423944
4	[3 7 8 2 1 5 4 6]	28788	0,0000347355
5	[3 6 2 7 1 5 4 8]	27871	0,0000358783
6	[3 2 5 1 8 6 7 4]	31776	0,0000314693
7	[3 8 7 1 4 6 2 5]	28737	0,0000347971
8	[3 4 1 5 8 7 2 6]	30758	0,0000325108
9	[3 2 1 7 5 4 8 6]	28982	0,0000345030
10	[3 5 4 7 6 1 8 2]	28596	0,0000349687

Then normalization of fitness is carried out on each individual in the initial population as shown in Table 5.

Table 5. Initial Population Normalized Fitness

#	Chromosome	Fitness	Normalized Fitness
1	[3 4 7 1 2 5 8 6]	0,00003184	0,0913692
2	[3 8 4 2 5 1 6 7]	0,00003542	0,1016537
3	[3 7 4 5 6 2 1 8]	0,00004239	0,1216371
4	[3 7 8 2 1 5 4 6]	0,00003473	0,099662
5	[3 6 2 7 1 5 4 8]	0,00003587	0,102941
6	[3 2 5 1 8 6 7 4]	0,00003146	0,0902910
7	[3 8 7 1 4 6 2 5]	0,00003479	0,0998391
8	[3 4 1 5 8 7 2 6]	0,00003251	0,0932793
9	[3 2 1 7 5 4 8 6]	0,00003450	0,0989952
10	[3 5 4 7 6 1 8 2]	0,00003496	0,1003314

Next, recombination operations are performed in the form of crossover and mutation. Crossover and mutation are carried out by selecting individuals in the population using the roulette wheel selection. The cumulative fitness value is calculated from the normalized fitness

value, then a random number is generated to determine which individual is selected based on the random number. This process can be seen in Figure 12 below.

Individual	Chromosome	Normalized Fitness	Cumulative Fitness	Bilangan Acak :	Selected Individual
1	[3 4 7 1 2 5 8 6]	0,0913692838	0,0913692838	0,576967149	6
2	[3 8 4 2 5 1 6 7]	0,1016537910	0,1930230748		
3	[3 7 4 5 6 2 1 8]	0,1216371990	0,3146602738		
4	[3 7 8 2 1 5 4 6]	0,0996623103	0,4143225840		
5	[3 6 2 7 1 5 4 8]	0,1029412403	0,5172638244		
6	[3 2 5 1 8 6 7 4]	0,0902910360	0,6075548603		
7	[3 8 7 1 4 6 2 5]	0,0998391764	0,7073940367		
8	[3 4 1 5 8 7 2 6]	0,0932793085	0,8006733452		
9	[3 2 1 7 5 4 8 6]	0,0989952127	0,8996685579		
10	[3 5 4 7 6 1 8 2]	0,1003314421	1,0000000000		

Figure 12. Roulette Wheel Selection

Crossover is done by selecting two individuals in the population. Suppose the selected individual is the individual in Table 6 as follows.

Table 6. Crossover Parent

Parent	Individual Number	Chromosome
P1	6	[3 2 5 1 8 6 7 4]
P2	4	[3 7 8 2 1 5 4 6]

Then the cut point value is generated randomly, for example in this test the cut point value is generated, which is 3. Then the gene value on the parent P1 chromosome is taken to the cut point, then the gene from parent P2 is entered from left to right with a note that it does not take the existing gene. This process can be seen in Figure 13.

P1	P2
[3 2 5 1 8 6 7 4]	[3 7 8 2 1 5 4 6]

Crossover		Chromosome Merging	Parent 2 Chromosome	Information
Step	1	[3 2 5 x x x x x]	[3 7 8 2 1 5 4 6]	Gene exist
	2	[3 2 5 7 x x x x]	[x 7 8 2 1 5 4 6]	Insert Gene
	3	[3 2 5 7 8 x x x]	[x x 8 2 1 5 4 6]	Insert Gene
	4	[3 2 5 7 8 x x x]	[x x x 2 1 5 4 6]	Gene exist
	5	[3 2 5 7 8 1 x x]	[x x x x 1 5 4 6]	Insert Gene
	6	[3 2 5 7 8 1 x x]	[x x x x x 5 4 6]	Gene exist
	7	[3 2 5 7 8 1 4 x]	[x x x x x x 4 6]	Insert Gene
	8	[3 2 5 7 8 1 4 6]	[x x x x x x x 6]	Insert Gene

Individual C1	[3 2 5 7 8 1 4 6]

Figure 13. Crossover Process

Mutations are carried out by exchanging genes in one individual. For example, individual C1 with chromosome [3 2 5 7 8 1 4 6] will be subjected to mutation. First, the mutation rate value will be determined, in this test the mutation rate value of 0.3 is used. Then a random number is generated, if the generated random number is smaller than the mutation rate value, then a mutation is carried out on the individual. Two exchange points were selected

randomly. For example, in individual C1 the exchange point is selected at positions 2 and 5, then a mutation is applied so that individual C1 has a chromosome [3 8 5 7 2 1 4 6].

The next generation will be obtained by entering the crossover and mutation results into the next population. Suppose the population size is 10 individuals, then crossover and mutation are carried out until 10 new individuals are obtained. The results of this second generation of individuals can be seen in Table 7 below.

Table 7. Second Generation Population

#	Chromosome
1	[3 7 4 2 1 5 8 6]
2	[3 2 5 7 4 8 1 6]
4	[3 2 5 1 8 6 7 4]
5	[3 5 4 7 6 8 2 1]
6	[3 4 1 5 6 7 2 8]
7	[3 2 1 7 4 8 5 6]
8	[3 5 4 6 1 2 8 7]
9	[3 2 5 4 7 1 8 6]
10	[3 4 7 5 6 2 1 8]

These individuals will then repeat the process carried out by the initial population individuals starting with calculating their fitness values. This process is continuously carried out until the termination condition is reached. The best individual obtained is the solution of the route optimization case. Suppose there are individuals with chromosomes [3 5 7 4 8 1 6 2], then the chromosome decoding is carried out again by adjusting each gene back to the location name based on the gene adjustment table so that a solution is obtained as shown in Figure 14 as follows.

#	Location Name
1	Rumah Radakng
2	RS Antonius
3	Pasar Flamboyan
4	Ayani Megamall
5	RS Yarsi
6	Gereja Katedral Santo Yosef
7	Universitas Tanjungpura
8	Hotel Orchardz Ayani

Chromosome
[3 5 7 4 8 1 6 2]
Solution
Pasar Flamboyan → RS Yarsi → Universitas Tanjungpura → Ayani Megamall → Hotel Orchardz Ayani → Rumah Radakng → Gereja Katedral Santo Yosef → RS Antonius

Figure 14. Solution Decoding

Black-box testing is intended to test the functionality of the designed software. This test is carried out to find faults in functions, interfaces, data structures, performance, and terminations. By testing various test scenarios, it can be found if the software works according to the expected results. There are 11 test scenarios tested, namely accessing the optimization page, locations appearing in the search location, locations can be added to the visit list, locations can be displayed on the map, removing a location from the visit list, displaying a selection of starting points from the list visits, displays the optimization processing page, displays the process of changing the best routes in the optimization process, displays the optimization result route, displays the optimization result route in google maps, and saves the

optimization result route. Of the 11 scenarios tested, no problems were found so that the test results were as expected.

White-box testing is carried out with the aim of checking the results based on the internal structure and flow of the software code. The white-box testing in this study focuses on the genetic algorithm component of the software. White-box testing is carried out on the components of the genetic algorithm used, namely the crossover, mutation, and selection processes.

In crossover unit testing, input is required in the form of two individual parents which will produce a new individual as a result of the crossover.

Parameter	Individu orderA, individu orderB
Line	**Coding**
Inp	function crossOver(orderA, orderB) {
1	var start = 0;
2	var end = Math.floor(getRandomInt(start+1, orderA.length));
3	var neworder = orderA.slice(start, end);
4	for (var i = 0; i < orderB.length; i++) {
5	var node = orderB[i];
6	if (!neworder.includes(node)) {
7	neworder.push(node);
	}
	}
Out	return neworder;
	}

Figure 15. Crossover Code

Based on Figure 15 above, there is a loop and decision tree. The flow traversed by the system can be described as three independent paths that can be tested. The following in Table 8 is a flow that the system can go through.

Table 8. Crossover Independent Path

#	Path
1	In-1-2-3-4-Out
2	In-1-2-3-4-5-6-4-Out
3	In-1-2-3-4-5-6-7-4-Out

After obtaining the independent path in Table 8, unit testing is carried out with case examples made to test the path. Based on the tests carried out, path 1 and 2 have results that the system should not pass, so input validation is needed. While path 3 has no problems and runs as expected.

The next test to be carried out is mutation unit testing. In mutations unit testing, input is required in the form of an individual who will be mutated into a new individual resulting from the mutation.

Parameter	Individu order, double mutRate
Line	Coding
Inp	function mutation(order, mutRate){
1	var reversed = order.reverse();
2	for (var i = 0; i < totalNode; i++) {
3	if (Math.random(1) < mutRate) {
4	var indexA = Math.floor(Math.random() * (reversed.length-1));
5	var indexB = Math.floor(Math.random() * (reversed.length-1));
6	swap(reversed, indexA, indexB);
	}
	}
Out	order = reversed.reverse();
	}

Figure 16. Mutation Code

Based on Figure 16 above can be found a loop and a decision tree. The flow traversed by the system can be described as three independent paths that can be tested. The following in Table 9 is a flow that the system can go through.

Table 9. Mutation Independent Path

#	Path
1	In-1-2-Out
2	In-1-2-3-2-Out
3	In-1-2-3-4-5-6-2-Out

After obtaining the independent path in Table 9, unit testing is carried out with case examples made to test the path. Based on the tests carried out, there were no problems found in the three paths, but in lanes 1 and 2 there was no change in individuals which this path should not have done. Therefore, it is necessary to validate the input to avoid paths 1 and 2.

In selection unit testing, input is required in the form of two population lists and a probability value that will return an individual from the selection.

Parameter	*Population list, double prob*
Line	*Coding*
Inp	function selection(list, prob){
1	var index = 0;
2	var r = Math.random(1);
3	while (r > 0){
4	r = r - prob[index];
5	index++;
	}
6	index--;
Out	return list[index].slice();
	}

Figure 17. Selection Code

Based on Figure 17 above, there is a loop and decision tree. The flow traversed by the system can be described as two independent paths that can be tested. The following is a flow that the system can go through.

Table 10. Selection Independent Path

#	Path
1	In-1-2-3-6-Out
2	In-1-2-3-4-5-4-5-Out

After obtaining the independent path as in Table 10, unit testing is carried out with case examples made to test the path. Based on the tests carried out, path 1 has results that the system should not pass, so input validation is required. While path 2 has no problems and runs as expected.

Time test is carried out with the aim of getting the average time required for the genetic algorithm to solve the problem in the test case. The test is carried out by running the test case 10 times and taking the average time when the genetic algorithm finds the best solution that has been tested by the brute force method. The time calculation starts from initialization of the initial population until the genetic algorithm gets the best solution. Here are the results obtained in Table 11, Table 12, and Table 13.

Table 11. Time Test (6 Points of Location)

#	Time (Seconds)	#	Time (Seconds)
1	0,046	6	0,056
2	0, 053	7	0,020
3	0,031	8	0,091
4	0,197	9	0,068
5	0,025	10	0,051
Average Time		0,065	

Table 12. Time Test (8 Points of Location)

#	Time (Seconds)	#	Time (Seconds)
1	1,723	6	0,649
2	0, 309	7	1,368
3	3,025	8	1,074
4	0,105	9	3,572
5	0,259	10	1,073
Average Time		1,428	

Table 13. Time Test (10 Points of Location)

#	Time (Seconds)	#	Time (Seconds)
1	32,867	6	18,937
2	120,639	7	1,368
3	17,163	8	10,171
4	4,521	9	51,257
5	55,481	10	85,427
Average Time		39,7831	

After 10 trials for 3 different cases, the results can be seen in table 11, table 12, and table 13 for cases with a total of 6 locations, 8 locations, and 10 locations having an average time of 0,065 seconds, 1,428 seconds, and 39,7831 seconds. For 6 to 8 locations, the calculation results are quite fast, while for 10 locations the calculation results have a very wide range with the comparison of the fastest and the latest times being quite far.

The results of the research demonstrated that the genetic algorithm approach achieved notable improvements in route optimization, particularly in scenarios involving multiple waypoints or destinations. The algorithm effectively minimized travel distances and reduced the overall time required for reaching all the destinations, leading to enhanced efficiency and cost savings. Moreover, the algorithm's ability to adapt and refine its solutions over multiple generations made it highly adaptable to changing conditions and constraints.

Furthermore, the integration of the Google Maps API provided accurate and real-time data on traffic conditions, road networks, and distance calculations, allowing the genetic algorithm to generate routes that were both optimal and feasible in real-world scenarios. The combination of these technologies proved to be a powerful tool for solving complex routing problems in various domains, including transportation logistics, delivery services, and urban planning.

4. CONCLUSION

The conducted research has successfully developed a route optimization software utilizing genetic algorithms and the web-based Google Maps API. The software encompasses essential features such as location and route management, along with the optimization of location points into an optimal route. The research outcomes indicate that the genetic algorithm employed in the software can efficiently determine optimal solutions for scenarios involving fewer than 10 locations, with a satisfactory average time. However, it should be noted that the best and worst time spans at 10 points exhibit significant variation. This disparity arises due to the utilization of random number generation throughout the genetic algorithm calculations. Random number generation is employed during the initial population generation, selection using probability fitness, and gene selection in crossover and mutation processes. Consequently, these random variations impact the number of generations and subsequently result in different completion times, despite the identical problem being addressed. Despite the variations, the research affirms the software's capability to provide effective route optimization solutions, highlighting the potential benefits of employing genetic algorithms and the Google Maps API in solving similar problems.

Apart from the inconsistency of completion time, this study also has limitations in the form of a maximum number of location points, which is 10 points, and the resulting route is only limited to the path that can be passed by four-wheeled vehicles. It is worth mentioning that the genetic algorithm employed in this study operates based on the pre-designed component model. The software components, including those beyond the genetic algorithms, have been implemented and function as intended according to the model. These components complement the genetic algorithm in facilitating effective route optimization.

5. SUGGESTED

In addition to addressing the time inconsistencies in finding solutions, an important avenue for improvement lies in developing a route optimization feature that incorporates both cost and time factors. By combining these crucial factors, the software can achieve improved

performance and deliver more valuable outcomes. This enhancement would empower users to make informed decisions, balancing efficiency and cost-effectiveness in their route planning endeavors.

6. REFERENCES

[1] David, "Artificial Intelligence sebagai Solusi Menghadapi Era Disrupsi Digital 4.0", JUDIMAS, Vol. 1, No. 1, pp. 107-116, 2022.

[2] Crepinsek M., Liu S. H., and Mernik M., "Exploration and Exploitation in Evolutionary Algorithms: A Survey", ACM Comput. Surv., Vol 45, No. 3, pp. 1-33, 2013.

[3] Reddy M. J., Kumar D. N., "Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: a state-of-the-art review", H2Open J, Vol. 3 No. 1, pp. 135–188, 2020

[4] Muftikhali Q. E., Yudhistira, A. Y. F. D., Kusumawati A., dan Hidayat S., "Optimasi Algoritma Genetika dalam Menentukan Rute Optimal Topologi Cincin pada Wide Area Network", JIM, Vol. 13, No. 1, pp. 43-48, 2018.

[5] Kramer O., Genetic Algorithm Essentials, Germany: Springer International Publishing, 2018.

[6] Kuway S. M., dan Yuliani I. D. A. E., "Pengembangan Aplikasi Pengelolaan Alat Tulis Kantor (ATK) Berbasis Web", KNSI 2016, pp. 1269-1276.

[7] Solichin A., Pemrograman Web dengan PHP dan MySQL, Jakarta: Penerbit Budi Luhur, 2016.

[8] Prasetyo A. C., Arnandi M. P., Hudnanto H. S., dan Setiaji B., "Perbandingan Algoritma Astar dan Dijkistra dalam Menentukan Rute Terdekat", SISFOTENIKA, Vol. 9, No. 1, pp. 36-46, 2019.

[9] David, Widayanti T., dan Khairuzzahman M. Q., "Performance Comparison of Cat Swarm Optimization and Genetic Algorithm on Optimizing Functions", ICORIS 2019, pp. 35-39.

[10] Kosasi S., "Optimasi Pemilihan Rute Transportasi Dengan Algoritma Simulated Annealing", TRIFORMATIKA, Vol. 4, No. 1, pp. 18-25, 2014.

[11] Utami P. Y., Suhery C., dan Ilhamsyah, "Aplikasi Pencarian Rute Terpendek Menggunakan Algoritma Genetika", CODING, Vol. 2, No. 1, pp. 19-25, 2014.

[12] Santoso H., dan Sanuri R., "Implementasi Algoritma Genetika dan Google Maps API Dalam Penyelesaian Traveling Salesman Problem with Time Window (TSP-TW) Pada Penjadwalan Rute Perjalanan Divisi Pemasaran STMIK El Rahma", TEKNIKA, Vol. 8, No. 2, pp. 110-118, 2019.

[13] Prakasa B. S., Istiadi, dan Priyandoko G., "Optimasi Rute Penjemputan Penumpang Travel Menggunakan Algoritma Genetika Berdasarkan Tingkat Kemacetan Lalu Lintas", CIASTECH 2019, pp. 395-402.

[14] Sugiyono, Metode Penelitian Kuantitatif, Kualitatif, dan R&D, Bandung: Alfabeta, 2013.

[15] Pressman R. S., dan Maxim B. R., Software Engineering: A Practitioner's Approach, 9th Edition, New York: McGraw-Hill Education, 2019.

[16] Chowdhary K. R., Fundamentals of Artificial Intelligence, New Delhi: Springer, 2020.

[17] Yu X., dan Gen M., Introduction to Evolutionary Algorithms, London: Springer, 2010.

[18] Dincer A., dan Uraz B., Google Maps Javascript API Cookbook, Birmingham: Packt Publishing, 2013.
[19] Kramer O., Genetic Algorithm Essentials, , Germany: Springer International Publishing, 2017.
[20] Russell S. J., dan Norvig P., Artificial Intelligence: A Modern Approach, 4th Edition, New York: Pearson, 2021.