OPTIMAL DESIGN OF WASTE TRANSPORT SYSTEM USING GENETIC ALGORITHM

[1]Aurelia Ailyn Andoko, [2]Hironimus Leong
[1,2]Departement of Informatics Engineering Faculty of Computer Science, Soegijapranata Catholic University
[2]marlon.leong@unika.ac.id

ABSTRACT

Wastes are residues of daily human activities or solid natural processes. Over the years the buildup of waste has been gradually increasing, due to the increase of human population / human activities. From the waste that has been produced, the remaining waste usually is placed in a temporary shelters for waste or TPS. This TPS is then later taken to a final processing place for waste or called TPA, there needs to be an optimum route from one TPS to another TPS with the final destination TPA to prevent waste from overflowing in one TPS which can cause harm in the environment. This research uses genetic algorithm to optimizes route between one temporary waste shelters to another, since genetic algorithm is an optimization algorithm, this research will prove whether genetic algorithm can solve the problem or not, the algorithm will be compared with another optimization algorithm, which is brute force to determine is genetic algorithm really a good optimization algorithm. Genetic algorithm has proven to be more optimum from the original path and also from brute force algorithm, but genetic algorithm needs more time to compute and is considered to be more complex and hard to understand compared to the brute force.

Keywords: Genetic_algorithm, tsp, brute_force

INTRODUCTION

Background

Wastes are residues of daily human activities or solid natural processes. Over the years the buildup of waste has been gradually increasing, due to the increase of human population / human activities. The wastes that were produced is usually placed in a garbage bin and then be taken to the temporary shelters for waste or TPS. From the temporary shelters, the wastes will then be taken into the final processing place for waste or commonly called TPA. Because of the increasing waste being produces, the process of taking the waste from TPS into TPA also needed to be increased to prevent waste from overflowing in one TPS, which can cause disease and an unpleasant smell to the surrounding environment.

Genetic algorithm is a minimum or maximum unconstrained function search method that uses random selection processes, such as selection, crossover, and mutation to stimulate the next reproductive operator in an organism. Yang and Su (2007) has proven that the optimization model can reduce the sewer rehabilitation cost up to 20%. However, cost estimates were related to market research or empirical equations.

In this research, genetic algorithm is applied to the waste transport route to optimize the route, optimize in this research means to shortened the distance between TPS to TPS and also to find the most optimal place to build a TPA. Before executing the process, there will be a period to observe the places of TPS across Tangerang. Solutions will then undergo selection to form the next populations and will be crossover and mutated. After undergoing this operation, the end result will be compared with the original data to see whether the genetic algorithm optimization works well or not.

Problem Formulation

1. Will genetic algorithm be able to optimize the path of waste transport route?
2. Compared with another solution, can genetic algorithm perform better?

Scope

This study is only focusing on the route of the waste transport, and does not consider the maximum volume of the truck to pick up the waste. This study also only covers the region Tangerang and not anywhere else.

Objective

The purpose of this research is to find the most optimum route for waste transport, and to find out the most strategic place to build the final landfill so that the cost for fuel is more efficient when collecting waste from the temporary waste dump.

LITERATURE STUDY

In this research, Bhaskar and Maheswarapu [1] discussed the rapid growth of the power system structure that have marked open access. This has raised concerns about the likeability and the impact of thoughtful security failures during the recent blackouts around the world. Therefore, they tried to find the optimal power flow, using a reformed hybrid genetic algorithm (between binary and real coded genetic algorithm). Binary-encoded genetic algorithm inherits the advantage that the decision variables are encoded in strings of finite length, and it is easier to implement and visualize by exchanging the parts of two parent strings. On the other hand, real coded genetic algorithm has the advantage that the actual parameters can be used as is, and the crossover and mutation are applied straight to the actual parameter values. Their research was tested on their C2D computer's standard IEEE-30 bus system with a 2.1 GHz switching speed. Later on, they compare the proposed genetic algorithm with adaptive genetic algorithm, to see which works best to find the optimal power flow. When compared with the variants of genetic algorithm such as the standard genetic algorithm, the adaptive genetic algorithm, particle swarm optimization, and differential evolution, the hybrid method has the lowest fuel cost and second fastest algorithm. This article shows that for finding an optimal power flow, a hybrid of genetic algorithm works even better but slower than basic genetic algorithm.

This research by Rochman Rendiyatna [2] is a development of previous research result that is done by Cepi Dea Iskandar, back in 2013. The reason why this research is being developed is

because the previous research only uses Heuristic Dispatching Rules method, and does not compare this method with any other method. From the previous research it was found that the scheduling of CV Boeing Teknik Mandiri with the Heuristic Dispatching Rules method can save the production time by 3%. This research uses the Genetic Algorithm method to minimalize the completion time and to compare the efficiency of Genetic algorithm with Heuristic Dispatching Rules. In conclusion GA method result better compared to the Heuristic Dispatching Rules with the efficiency of 5,25%. From this research, we can see that we need another method that works as a comparison to actually conclude that the method we are using is indeed more suitable/efficient.

Aprilia and Wayan [3] discuss The meals that is being fed on with the aid of using affected person is one of the elements that have an effect on blood strain on hypertensive patients, if they're not being careful can cause a growing risky disease, one manner to manage on hypertensive patient may be achieved with the aid of eating meals with much less salt. Therefore, in order to control the blood pressure of hypertensive patients, the person need adequate nutrition and a diet with low salt. To make it easier for hypertensive patients, the composition of foods can be calculated with software. The results demonstrated an optimized search for the composition of foods to approximate the nutritional needs of hypertensive patients. In this research, a hybrid method of Genetic Algorithm and Variable Neighbourhood Search is used to solve food composition problems. From this research, it can be concluded that they successfully solve optimization of food composition problem by using the hybridization of GA and VNS compared to pure GA or pure VNS, although the hybrid method need a longer computational time compared to the pure GA and VNS. In this research, it also shows that the hybrid of genetic algorithm works better compared to the basics, but it need longer time to operate.

Yusop et al. [4] discuss Gestational Diabetes Mellitus or commonly called GDM is one of the most common diseases of pregnancy, specifically with high prevalence in pregnant women. Diabetes mellitus as an atypical and a complex illness that require the person affected to take an excellent care in their selves, via way of means of retaining continual hospital treatment of the glycemic control. In recent years, many Gestational Diabetes Mellitus patients keep track of their blood sugar levels and also practice the traditional method of recording their food intake. When compared with the modern mobile health monitoring technologies that are already available, the traditional practices are not efficient. The purpose of this research is to develop a GDM monitoring application for patients and their physicians. Constructing a predict algorithm for meal intake recommendation is necessary to maintain blood sugar levels. The software needs to know the patient history, biosignals, and electronic diary records. It is also important to know the patient's diet, sleep time, physical activities, their insulin injections and also the blood glucose readings. Genetic Algorithms have been found to help achieve optimal results as they are effective and robust algorithms that help to find the optimal solution under many conditions. This is good because the predicted blood level needs to be accurate and also optimal to assist the users. This study uses the food information divided into several types. The software gives a platform for

diabetes patients to perform self-monitoring based on their dietary intakes. On the down side, the software is mainly made for Malaysians, this is because of the meals that the software recommends mainly uses Malaysians' local foods. From this research, we can see that Genetic Algorithm serves a precise prediction to find the blood glucose level, this research can be implemented later on to see the precision of genetic algorithm.

Armandi et al. [5] stated that because the resident number in Yogyakarta keeps increasing every year, it has implications for increasing consumption activity as well as waste generated. In 2016 they observed that every vehicle that is dispatched in Yogyakarta to collect trash from the same location to the point that they are supposed to come back with overload trash capacity every time. There has been previous research about this topic, using the saving matrix method in Palembang, also using Vehicle Routing Problem model which were completed by using the hybrid of genetic algorithm and simulated annealing in Denpasar. In this research, Armandi et al. uses a hybrid of genetic algorithm and local search, this method is used to finish the evaluation process from the previous Vehicle Routing Problem that has been developed. The limitation from VRP are the vehicles that are being used has different capacities (heterogeneous fleet), the garbage dump point as intermediate facility that has to be visited by vehicle before starting their route and before returning, condition that lets a vehicle to create a new route (multiple trips), and lastly a condition that lets a point service to get the service more than once by the next vehicle (split delivery). This research uses the route of garbage collecting vehicles as their dataset. Hybrid genetic algorithm that is used in this research uses split technique by Baker & Ayechey (2003) whereas the other steps such selection, crossover, and mutation uses the Roulette Wheel's Selection, Multiplepoint Crossover, and Scramble Mutation with the parameter that is used is 100 populations and 25000 iterations. This research then compared the hybrid genetic algorithm with Particle Swarm Optimizations with the same parameter, and after doing all of the process, they have concluded that this research has successfully develop the VRP model to solve the Yogyakarta waste transportation route considering many limitations, and the hybrid of genetic algorithm proofs to be more effective and efficient compared to Particle Swarm Optimization. From this research, I think that using PSO as a comparison can be implemented in the future, also the method for steps (selection, crossover, mutation) are stated and can be researched more to be implemented in the future.

Sundah[6] discuss that one of the power plants in North Sulawesi is powered by a diesel generator. There are 2 issues, the first one is the unit commitment, which determines the scheduling of creating units to reduce the system fuel cost. Second is the economic distribution, a key process that determines the performance produces by each unit. To meet a specific load generator with the aim of reducing operating costs. Genetic algorithm methods under development can correctly handle problems relating to unit-joining operation uncertainties. The data that is being used is the diesel power plant of North Sulawesi. In this research, genetic algorithm is being implemented by doing 4 steps, which is developing a population consisting of strings, evaluating each string, picking the process that gets the best string and genetically engineer it to make a new population

of strings. This study has developed a well working genetic algorithm program for distribution optimization and analysis at power plants. The planning performed by using genetic algorithm method can reduce the production costs up to 6.64% per day. From this research, genetic algorithm has done a great job to decrease cost and also solve problem that has been stated before, but there are no comparison with other method, so we can only conclude that genetic algorithm is optimal, but further comparison must be done to test whether genetic algorithm is still the best method or not to use.

Puspitasasri et al. [7] discuss that heart disease is where the condition of a heart cannot function normally, or often called an uncompensated state, blood circulation which is not normal can cause shortness of breath, fatigue, and pain in the heart. Abnormalities in kidney, liver, and blood pressure and sodium resorption resulting in edema, stated by Antonius Widoyoko (2011). Inside the heart there is a coronary heart disease, which is a very important heart disease, this is because this disease happens to everyone and is considered as the main cause for death in several countries including Indonesia. Someone suffers from coronary heart disease if the blood flow to its heart is blocked by fat, three main risk factor that has caused coronary heart disease are smoking habits, lack of physical activities, and unbalanced meal. In fulfilling nutrition in patients the human heart is faced with a situation where we have to fulfill nutrition without being annoyed and triggers it to get worse heart condition, hence why this research is being conducted. This research uses evolutionary algorithms, which is an optimization technique that copies the biology evolution process, it also uses real-coded genetic algorithm. The process of genetic algorithm starts with creating random individuals that have certain chromosomal gene arrangement, and then the other method, such as crossover uses extended intermediate crossover, for mutation this research uses random mutation and then evaluation and selection. The dataset that is being used in this research are the weight, height, age, and gender of coronary heart patient and also 271 foods with the content of carbohydrate source, protein source, vegetables, fruits, snacs, and oil fat. From this research a conclusion can be concluded as real coded genetic algorithm can be implemented to optimize problems food composition for patients by giving a solution in the form of a combination recommendation food for the patients, they also concluded that the size of population and generation also effects the value of fitness, if the size population and generation are few then the search is getting narrow, this causes convergence.

Yang and Su [8] discuss that sewerage is an important element of a city, but it is also often neglected because of the difficulties to maintain, monitor and also rehabilitate when the sewerage is underground. However, because of corrosive wastewater, cracks and also leaks may occur. Carelessness regarding these problems has significantly raises the cost of maintenance and also rehabilitation. So, to stop even worse failures, it is necessary to do a regular rehabilitation. In this study, trade-off problem is a multi-goal optimization to reduce the price of rehabilitation and also to escalate the service life. Solving the time-cost trade-off problem, this research uses genetic algorithm combined with pareto curve and also machine learning. In this research, researcher uses the sewer system of the 15th district of Kaohsiung City in Taiwan as the study site, where the

sewer system is about 0.5 km and services 12,000 people. The initial step in this study is to preprocess the data, and after the chromosome-coding preprocessing is done, the chromosomes are encoded by integers representing two decision variables. This includes rehabilitation procedures and alternative materials. The next step is to generate the initial population, where genetic algorithm create an initial population randomly, that represents the number of possible solutions. After that, estimation of total rehabilitation cost is done, next is estimation of rehabilitation efficiency and then continued with computation of fitness, next is generation of the next population followed by crossover operator, then mutation operator, and lastly termination criterion. From the inspection records of Kaohsiung City sewer system, 59 out of 63 pipes in the sewage is defective, with 1 of 3 alternative repair methods of 1 of 4 alternative replacement materials, had to evaluate. Because of the difficulties of the sewerage structures, the sewerage retrofit trade off problem is a large scale optimization problem. Optimization based on genetic algorithm has been proven to create an optimal rehabilitation plans. The best models reduce rehabilitation costs by 20% compared to traditional expert estimates. This research can be a good reference to see how they can implement a hybrid genetic algorithm to improve the sewage rehabilitation considering there are many complexity from the sewer infrastructure.

Vairavamoorthy and Ali [9] The goal of the layout of water distributions structures is to deliver water with enough portions at normal pressures to the consumers. Because of the difficulties frequently encountered with inside the layout of the water distribution structures, springing up from the range of layout additives and their interactions, the conventional methods of trials and error and “rules of thumb” are all time consuming and likely to create an answer that are suboptimal. In this study, the researcher has presented a genetic-based method that applies true encoding to variables and overcomes problems that related to redundancy that occur when using the binary alphabetic encoding. The problems on this study are redundant values generated whilst the use of binary alphabet and large computational attempt related to the use of hydraulic solver for each string of population. This research uses the parallel expansion of New York City tunnel for an example as well as Hanoi network. Real coded genetic algorithm coupled with Linear Transformation Function is proposed to find the most optimal design of water distribution system. This method has been tested and has shown to be efficient and also robust. The proposed method also produces good results quickly.

Xin and Hongxia[10] discuss the main issue of completing trains is to assign the thickness of the exit with purpose and to decide the rolling for and relative convexity. A modified genetic load-sharing model algorithm creates a mathematical model of the complete train, taking into account the shape, thickness, and also the load balance. Because of genetic algorithm’s drawback, a new genetic algorithm has been made to improve and applied to the model. From the study it is proven that the improved genetic algorithm is more superior. Compared with the standard genetic algorithm, the improved GA is much more advanced when being faced with complex issues. From this research the method provide information that the improved genetic algorithm is better than the standard that have some significant disadvantages.

RESEARCH METHODOLOGY

Data collection

Data will be collected from 'Dinas Lingkungan Hidup Kota Tangerang'. This data provide the data of temporary waste location with 212 data. Those data will then be collected for this research. The information that will be taken is the coordinates of the locations.

Data pre-processing

From the 'Dinas Lingkungan Hidup Kota Tangerang' the location and also the coordinates of each TPS can be found. From this, a node representing each TPS can be made. While the data collected also contained some additional information such as the address of each TPS, the districts and also the wards, those component will not be used further in this research.

Implementation

The implementation in this step requires multiple processes. Before initializing the first step there will be a process of fitness evaluation, the drainage system that doesn't seem fit for the process will be eliminated. The first step is selection, selection is the process where from the initial population, parents will be selected, this has the purpose to randomly pair up the most fit individuals with other parents.

Next step is crossover, crossover operator is applied to create better strings, the function of this step is to make clones of good strings but does not create new ones. Crossover will create offstrings, this is done by taking the bits from the one parent and another parent, after the process of selection has already happened. After that there will be mutation, mutation subjects the strings to mutation. It facilitates a change in genes within a chromosome. This step guarantees the search algorithm will not be trapped on a local optimum.

Evaluation

After the data has been processed, a path of each TPS of Tangerang's can be viewed. Evaluating the data has the purpose to determine the optimization of genetic algorithms in the waste transport route. A comparison will be conducted between the route from the result of genetic algorithm and also the original route based on the number of TPS registered, genetic algorithm will also be compared with another optimization algorithm which is Brute Force, to see which produce the more optimum result. The length of each pipeline will be calculated and that length will determine if genetic algorithm gives a more optimal length or not.

ANALYSIS AND DESIGN

Analysis

Genetic algorithm is widely known as an optimization algorithm with natural selections. Genetic algorithm in this research will be used to optimize the waste transportation route in Tangerang city. The route will be described as a traveling salesman problem or TSP; here genetic algorithm will find a solution to the traveling salesman problem, by finding the shortest path between each TPS. The goal of this problem is to find the most optimal route and the final destination being the first destination.

Pre-processing Dataset

Data that has been taken from the website will be processed so it is more suitable for the optimizing phase. Since the original data only shows the table of coordinates, there is a process of making nodes from those coordinates alone.

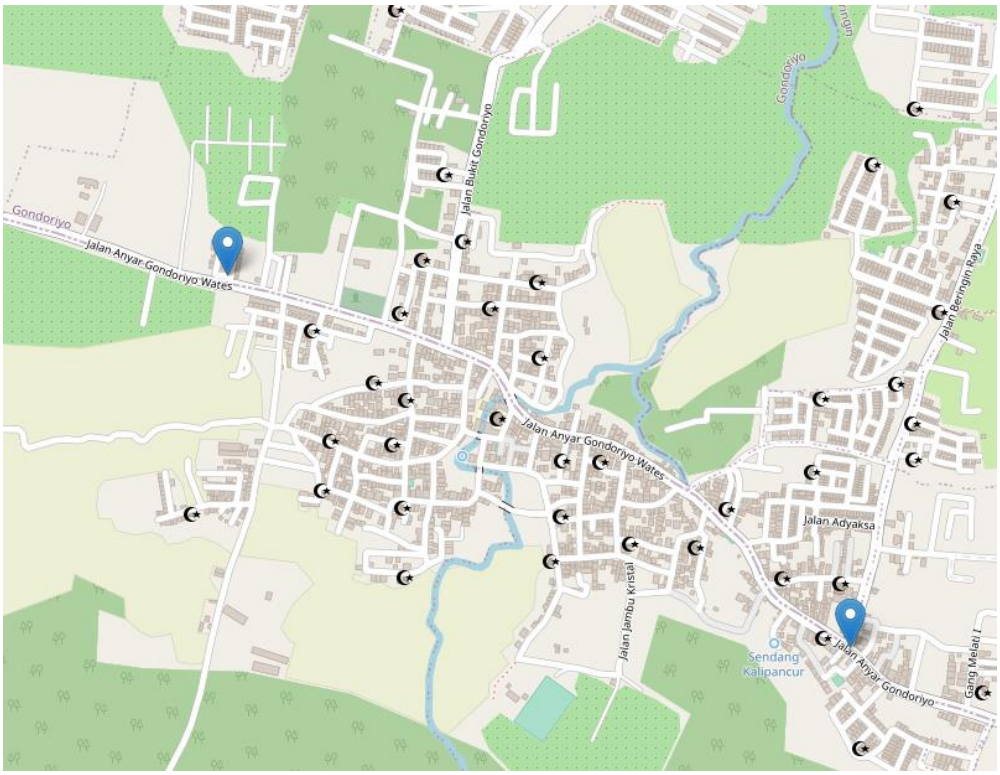

Figure 1. Nodes from Data

From here onward a dataset has been created, the dataset consist of the number of node, the longitude and also the latitude. The total dataset is 212. Table 5.1 is the preview of the dataset.

Table 1. Table 1 Dataset Examples

index	Y	X

0	-6.189553	106.638313
1	-6.191069	106.633267
2	-6.184849	106.648847
3	-6.174996	106.650157
4	-6.189553	106.638313
…	…	…
212	-6.225502	106.637648

Processing Data

After creating the dataset, path between nodes can be created. This path is used to only for visualization on how messy the route is without genetic algorithm. The path created is based on the nodes number, not by the nearest node available; this is why the path is messy and unorganized.

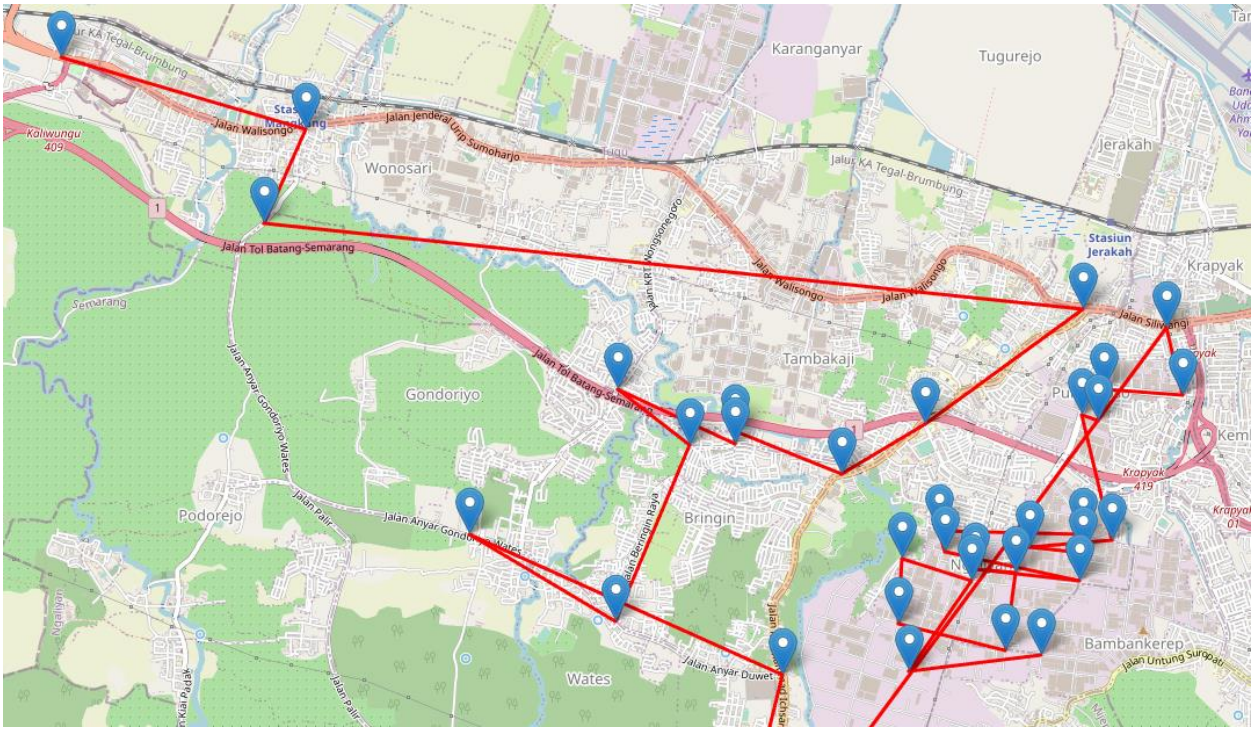

Figure 2. Original Path

The length of this path will also be calculated, this has the purpose to estimate the initial length and will it shortened after genetic algorithm has been conducted. To calculate the length from one node to another a formula called *Haversine Formula* will be used.

$$d = 2r\ arcsin\sqrt{sin^2(\frac{\varphi_2 - \varphi_1}{2}) + cos\varphi_1 * \cos\varphi_2 *\ sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}$$

d is the distance between two coordinates, r is the radius of the sphere, φ_2, φ_1 are the latitude from one point to another point, and the λ_2, λ_1 are the longitude from one point to another. An example from the data, calculating the distance between index [0] and index [1];

Table 2. Data Examples

index	X	Y
0	-6.96829	110.2848411
1	-6.973521	110.302996

From this data, the longitude and also latitude has been found, with the value;

$$\varphi_1, \lambda_1 = -6.96829, 110.2848411$$

$$\varphi_2, \lambda_2 = -6.973521, 110.302996$$

After that the latitude and also longitude can be inserted into the formula to be calculated, since r is the radius of the sphere, and in this case the sphere is the earth, the radius of the earth must be inserted, since the radius of the earth varies from 6356.752 km at the poles to 6378.137 km at the equator the one that will be used as r is 6371, next is to input the values into the formula and solve the equation.

$$d = 2r\ arcsin\sqrt{sin^2(\frac{-0.005231}{2}) + \cos(-6.96829) * \cos(-6.973521) *\ sin^2(\frac{0.0181549}{2})}$$

$$d = 2.0865244$$

With this formula the distance from one node to another node can be calculated, but to know the length of the overall pipeline, all of the distance must be added.

Genetic Algorithm

After getting the rough result of the route length, genetic algorithm will be used to solve the traveling salesman problem. The solution of genetic algorithm is represented by chromosome, this chromosome is made from a set of genes that represents solutions to unsolved problems. There are a few main process of genetic algorithm after initializing the parameter,

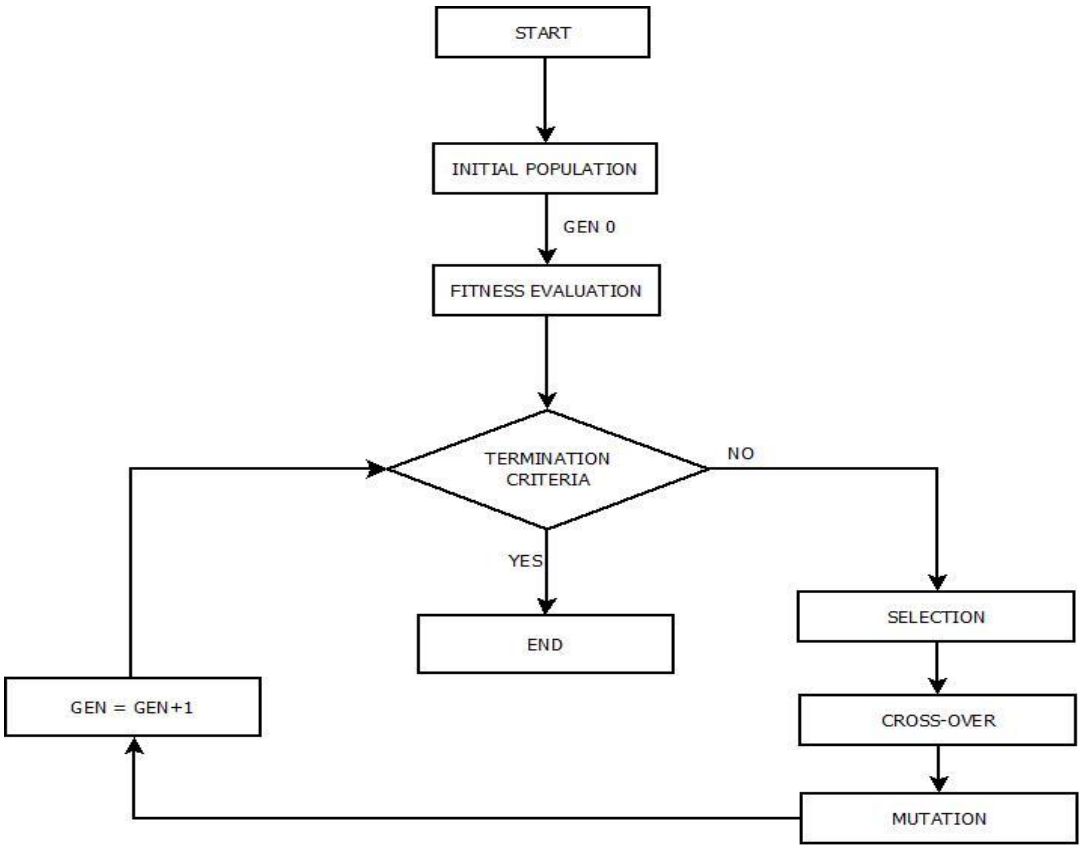

Figure 3. Genetic Algorithm Process

Initial Population

Initial population is the first step in the process of genetic algorithm, without the initial population the process cannot go forward; this is why this step is important. Population in genetic algorithm can be defined as a set of chromosomes. Here the set of genes is represented by strings.

The initial population in this research will be based from the index number of nodes that has been created in the beginning. In this research the index number consist from the 0 – 481, but for example index 0 – 40 will be displayed;

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40].

To create the population the first thing that's needed to do is to determine how much population is needed. In this research the number of population is 200, from here the original index number from 0 – 40 is considered as one chromosome, and the content of that chromosome is called gene, to get the initial population the gene of a chromosome will be randomized,

Table 3. Chromosome and Gene(s) Examples

Chromosome	Gene(s)
0	[30, 16, 9, 28, 4, 18, 29, 39, 15, 27, 26, 11, 31, 33, 13, 35, 25, 38, 5, 1, 32, 2, 22, 17, 40, 14, 7, 12, 20, 37, 6, 23, 36, 34, 10, 19, 21, 0, 8, 3, 24]

1	[22, 29, 26, 6, 15, 8, 35, 19, 9, 2, 12, 27, 34, 17, 20, 40, 5, 13, 30, 39, 31, 36, 24, 37, 25, 16, 3, 4, 11, 10, 23, 32, 21, 1, 33, 7, 18, 38, 0, 28, 14]
2	[0, 28, 5, 8, 34, 22, 6, 15, 39, 18, 30, 14, 17, 35, 37, 13, 3, 1, 7, 2, 26, 19, 31, 25, 4, 12, 38, 16, 40, 20, 11, 23, 24, 33, 9, 32, 29, 21, 27, 36, 10]
…	
200	[29, 27, 12, 20, 30, 2, 18, 35, 1, 14, 3, 16, 25, 6, 5, 8, 10, 37, 38, 0, 34, 36, 40, 4, 13, 7, 19, 17, 9, 22, 11, 28, 15, 26, 21, 33, 24, 23, 32, 31, 39]

This is only the example of the initial population; if the number of population is set to 200 then the gene of a chromosome will be randomized until 200 times.

Fitness Evaluation

The fitness evaluation has the function to determine whether an individual is fit for the process or not. If the individual is not able to compete with the other then it will be considered not fit for the job. In this research, first thing to do is to calculate the distance matrix of each node. By finding the Euclidean distance, it will give us the length of the line segment. The formula for Euclidean distance is;

$$d = \sqrt{[(x_2 - x_1)^2 + [(y_2 - y_1)^2]}$$

Where d is the distance between the coordinates, (x_1, y_1) are the coordinates of one point and $(x_{2,} y_2)$ are the coordinates of the other point. For example;

I_1 = (-6.96829, 110.2848411)

I_2 = (-6.973521, 110.302996)

$$d = \sqrt{[(-6.973521 - (-6.96829))^2 + [(110.302996 - 110.2848411)^2]}$$

$$d = 0{,}01889348$$

This calculation is done until the matrix is complete. From this matrix distance between nodes can be found. For example the distance between index [0] and index [1] is 0.018893484459198227. From here the fitness of the entire chromosome in the population can be calculated. After creating the distance matrix, the initial population will be calculated with the distance of the nodes, so it can create a fitness value. The next step is to determine the minimum fitness value. By setting the fitness value, the chromosome that does not fit can be eliminated.

Termination Criteria

The termination criteria could be used to stop the genetic algorithm process altogether or to continue to process, this is determined by some common criteria such as if a solution from the minimum criteria has been achieved, or the limit number of generation has been reached, or the computation time is too long, or maybe all of the above. If the termination criteria has not been met, then the process will continue into selection. For example, in this research the termination criteria have been limited to 1000 generations. So after generating the fitness 1000 times the process will stop.

Selection

Selection is one of three core processes of genetic algorithm. The purpose of this process is to select individuals from the current generations which will be used for the next one. Based on their fitness score, two pairs of individuals will be selected for the next process, which is crossover. Selection that will be used in this research is the tournament selection, based on the trial and error of the data; this specific selection method gives the best result and also shorter computational time compared to another method.

Based on the figure below, the result of tournament is proven to be producing the best fitness. All of the method uses the same parameter of 40 nodes, 200 population, 1000 generations, 0.6 crossover probability, and also 0.4 mutation probability.

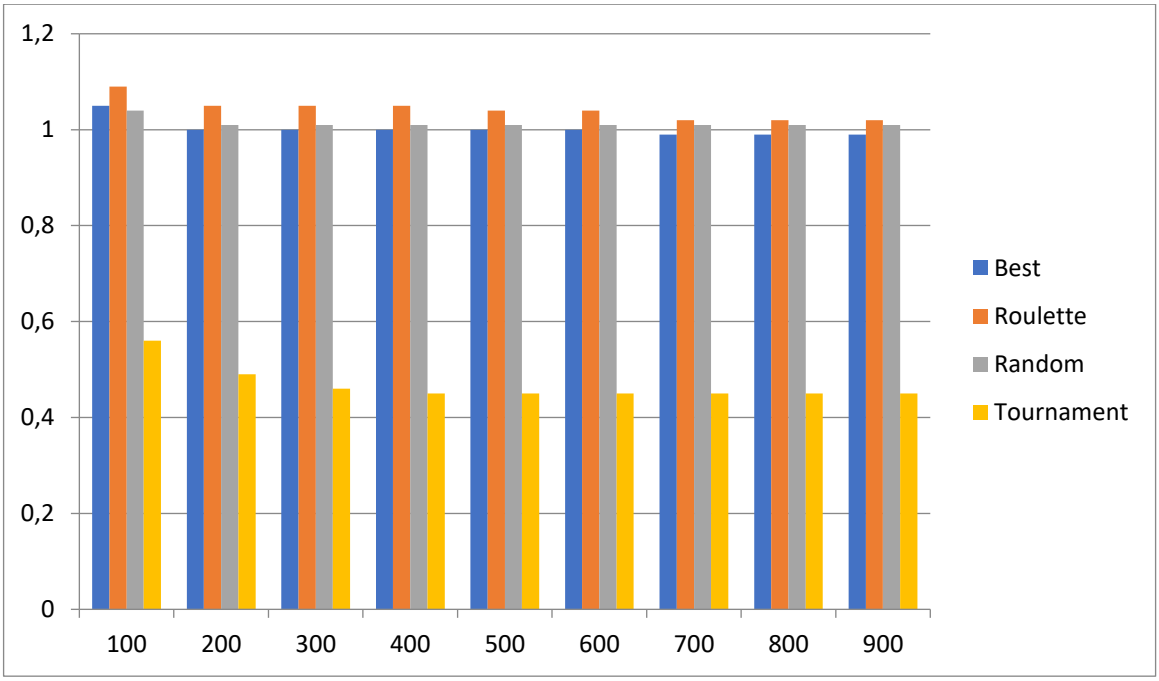

Figure 4. Selection Method Comparison

Crossover

Crossover is a process where the gene/s of a chromosome is swapped with another gene/s from a different chromosome; crossover will stop when the swapping process reached its crossover point. There is a crossover probability, that determines will a chromosome endure crossover or not, the common crossover probability is 60%, from here, a random number that range from 0.0 until 1 called *i* will be generated. *i* here will represent each chromosome available.

$$Crossover = IF(i < cross_{prob}; \text{true}; \text{false})$$

Where *i* is a random number between 0.0 until 1 and crossprob is the probability that has been determined, which is 80% / 0.8. Crossover probability is chosen based on trial and error from the data, since 0.8 provide the best result, this number is chosen, all of the trial have the same parameter of 40 nodes, 200 population, 1000 generations, and 0.4 mutation probability.

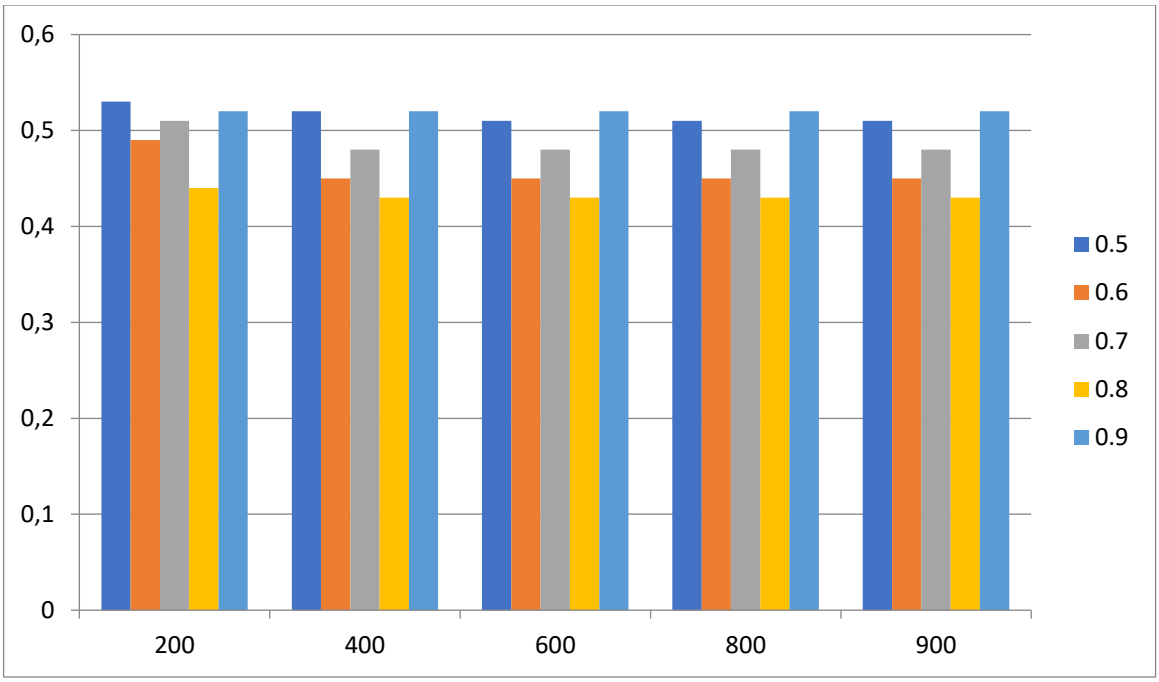

Figure 5. Crossover Probability Comparison

If value < crossover probability then crossover will happen, otherwise the crossover will not occur in that specific chromosome, for example;

Table 4. Crossover Probability

Chromosome	i (Random)	Crossover
0	0.9	No
1	0.3	Yes
2	0.9	No
…		
200	0.1	Yes

Next step is to actually do the crossover. The list of chromosome will be sorted by the fitness value, then for a chromosome that has higher *i* value then crossover probability, that chromosome will be crossover with another chromosome that fits the qualifications. A crossover point will be generated randomly; the function of this point is to stop the process of crossover, for example;

Table 5. Before and After Crossover Comparison

Before Crossover		
Chromosome	***i (Random)***	***Gene(s)***
1	0.3	[**22, 29, 26, 6, 15,** ||| 8, 35, 19, 9, 2, 12, 27, 34, 17, 20, 40, 5, 13, 30, 39, 31, 36, 24, 37, 25, 16, 3, 4, 11, 10, 23, 32, 21, 1, 33, 7, 18, 38, 0, 28, 14]
200	0.1	[**29, 27, 12, 20, 30**, ||| 2, 18, 35, 1, 14, 3, 16, 25, 6, 5, 8, 10, 37, 38, 0, 34, 36, 40, 4, 13, 7, 19, 17, 9, 22, 11, 28, 15, 26, 21, 33, 24, 23, 32, 31, 39]

After Crossover		
1	0.3	[**29, 27, 12, 20, 30,** 8, 35, 19, 9, 2, 12, 27, 34, 17, 20, 40, 5, 13, 30, 39, 31, 36, 24, 37, 25, 16, 3, 4, 11, 10, 23, 32, 21, 1, 33, 7, 18, 38, 0, 28, 14]
200	0.1	[**22, 29, 26, 6, 15,** 2, 18, 35, 1, 14, 3, 16, 25, 6, 5, 8, 10, 37, 38, 0, 34, 36, 40, 4, 13, 7, 19, 17, 9, 22, 11, 28, 15, 26, 21, 33, 24, 23, 32, 31, 39]

After doing this process, the chromosome will then go through mutation as the final process of the generation.

Mutation

Mutation is the process of tweaking gene in a chromosome; the function of this process is to create a new solution to the problem, in this research, mutation has the function to create a new path that maybe has not been found yet. Since mutation has many variants, the one that will be used in this research is the swap mutation, where two random genes will be swapped.

$$Mutation = IF(e < mutationprob; \text{true}; \text{false})$$

Same with crossover, mutation will also have its own mutation probability. The common mutation probability is 0.4 or 40%, also the same process with crossover, mutation will generate a random number in the range of 0.0 until 1 that represent each chromosome, called *e*. The mutation probability is chosen based on the best result from the trial and error, which will be shown below; the parameter is all the same, with 40 nodes, 1000 generations, and also 0.6 crossover probability.

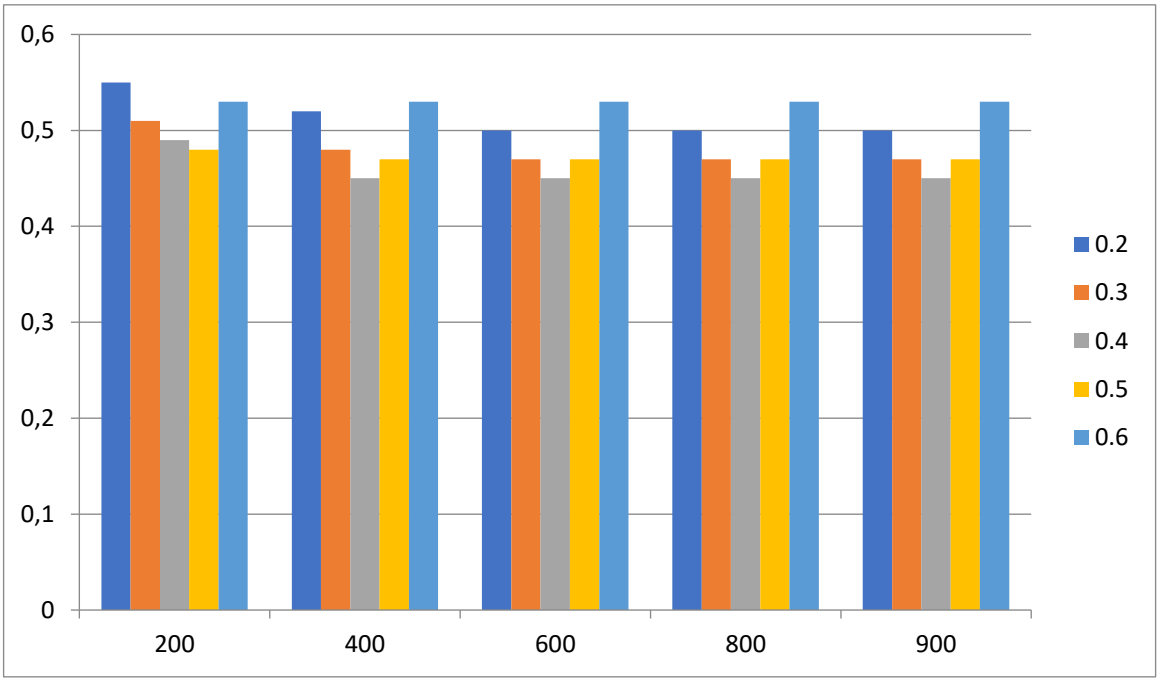

Figure 6. Mutation Probability Comparison

Table 6. Mutation Probability and Comparison

Chromosome	e(Random)	Mutation
1	0.1	Yes
200	0.5	No

Before Mutation		
Chromosome	***e (Random)***	***Gene(s)***
1	0.1	[29, 27, 12, 20, 30, 8, 35, **19**, 9, 2, 12, 27, 34, 17, 20, 40, 5, 13, 30, 39, 31, 36, 24, 37, 25, 16, 3, 4, **11**, 10, 23, 32, 21, 1, 33, 7, 18, 38, 0, 28, 14]
After Mutation		
1	0.1	[29, 27, 12, 20, 30, 8, 35, **11**, 9, 2, 12, 27, 34, 17, 20, 40, 5, 13, 30, 39, 31, 36, 24, 37, 25, 16, 3, 4, **19**, 10, 23, 32, 21, 1, 33, 7, 18, 38, 0, 28, 14]

After doing this process the chromosome will now get a new set of genes, this set of genes will be tested for its fitness value, if the fitness value turns out to be bad, then the result of the crossover and mutation will be deleted, otherwise the new chromosome will replace the previous best solution. This process is then repeated again and again until the maximum number of generations has been reached. After that, the best solution is obtained.

Evaluating Data

After the process of genetic algorithm has ended, an optimum route based on genetic algorithm has been produced, the route will be compared with brute force. Brute force can be counted as another optimization algorithm that solves problem in a simple way that relies on pure computational power and tries all possibilities rather than sophisticated techniques to improve the efficiency. For brute force to optimize a problem, it needs to find all the possible answers to produce the most optimum solutions.

The result of the path and also the distance of both algorithm will be shown, this has the purpose to compare both algorithm, which algorithm produces the most optimum route.

IMPLEMENTATION AND RESULTS

Implementation

```
1. m = folium.Map(location=[-6.9842650934006, 110.3832977495],
2.                zoom_start=15,
3.                min_zoom=14,
4.                max_zoom=18)
5.
6. for index, location in data.iterrows():
7.   loc=[location["x"], location["y"]]
8.   no=([location["no"]],[location["x"], location["y"]])
9.   rumah=folium.Marker(
10.        location=loc,
11.        popup=no
12.    )
13.    rumah.add_to(m)
```

```
14. m
```

Line 1-4 has the function of declaring m which is the variable that calls folium, which is a library that will help to build a map. Location in line 1 has the function to pinpoint the starting preview when the output of folium is displayed; zoom_starts is the default initial zoom level of the map, while min_zoom and max_zoom is the minimum and maximum value on how the map is allowed to zoom. Line 6 is the start of the loop, loop here has the function to display each node, and will stop when the maximum number of node has been reached. Line 7 is declaring loc equals to the coordinates x and y. Line 8, declare no, which contains the number of nodes, and coordinates x and y. rumah in line 9 has the function to display the marker using folium marker, location is the node or equal to loc and popup is the information that will pop out when the marker is clicked, popup contains no that has been declared before. After that rumah will be added into m in line 13, and line 14 is to display the m.

```
15. hitung=0
16. for index, location in data.iterrows():
17.   if hitung!=40:
18.     loc = [(data["x"][index], data["y"][index]),
19.           (data["x"][index+1], data["y"][index+1])]
20.     folium.PolyLine(loc,
21.                     color='red',
22.                     weight=3,
23.                     opacity=1).add_to(m)
24.   else:
25.     break
26.   hitung=hitung+1
27. m
```

Line 15 is to declare a new variable called hitung. Line 16 is to loop the data. Line 17 is the function to execute the true or false part of a condition, here the condition is hitung, if hitung is not 40 then the condition is considered true, line 18-19 is defining loc, here loc contains [x1, y1, x2, y2] this is used to draw the path between nodes. Line 20 is drawing the line using polyline, with the data loc; color, weight, and opacity is the modification of the path. And in line 23 the polyline will be added into m. If the hitung is equal to 40 then the condition is considered false and went into the else condition, line 25 is to break to terminate the loop, in line 26, hitung is added by one each time the loop occurs, and lastly in line 27 the map is displayed again.

```
28. count=0
29. pipelength=0
30. for index, location in data.iterrows():
31.   radius = 6371.0
32.   if count != 40:
33.     x1 = radians(data["x"][index])
34.     y1 = radians(data["y"][index])
35.     x2 = radians(data["x"][index+1])
36.     y2 = radians(data["y"][index+1])
37.     count=count+1
38.   else:
39.     x1 = radians(data["x"][index])
40.     y1 = radians(data["y"][index])
41.     x2 = radians(data["x"][index-40])
```

```
42.      y2 = radians(data["y"][index-40])
43.    xdistance = x2-x1
44.    xdistance = y2 - y1
45.
46.    p = sin(dx / 2)**2 + cos(x1) * cos(x2) * sin(dy / 2)**2
47.    count = 2 * atan2(sqrt(p), sqrt(1 - p))
48.
49.    distance = radius * count
50.    pipelength = pipelength + distance
51.
52.    print("Result",[index],": ", distance)
53.    print("Pipelength: ",pipelength)
```

Line 28, 29 is used to declare new variables, 30 is to start another loop. Line 31 is declaring R, R here is the radius of the earth in kilometer. Line 32 is another if else conditions that has the same function at line 17 before. Line 33 and 34 is to declare x1 and y1, which contains the radians from the coordinates. While the line 35 and 36 is the same but the coordinates will be from the next node available. Line 37 is to add the count variable, if the conditions in ifelse is not met, then the line 39-42 will take actions, there the process is almost the same with line 33-36 the difference between them is the x2 and y2 will take the radians from the first coordinates or index[0]. Line 43-44 is to calculate the distance between y2 and y1 / x2 and x1 in a new variable called dy and dx. Line 46-49 is the practice of haversine formula, line 50 is to count the overall pipelength, and line 52 and 53 is to print the result.

```
54. num_node=484
55.
56. coordinates=[]
57. for index, location in data.iterrows():
58.    x=data["x"][index]
59.    y=data["y"][index]
60.    coordinates.append([x,y])
61. names = [i for i in range(num_homes)]
62. coordinates_dict   =   {name:   coord   for   name,coord   in   zip(names,
    coordinates)}
63.
64. coordinates=np.array(coordinates)
65.
66. print(coordinates_dict)
```

Line 54 is to declare number of nodes, line 56 is to declare a new variable, and for line 57 is another loop, both the longitude and latitude will be inserted into variable x and y and will be merged together in the coordinates variable, line 61 is to declare names which contains the number of nodes, line 62 is declaring coordinates_dict, that contains names and also coordinate. Line 64 is to convert coordinates into array, and line 66 is to print the coordinates_dict.

```
67. distanceMatrix = distance.cdist(coordinates, coordinates, 'euclidean')
68. print(distanceMatrix)
69.
70. fig = plx.imshow(distanceMatrix)
71. fig.show()
```

Line 67 is to declare distanceMatrix, here the distance matrix is calculated using a library, the distanceMatrix will compute a distance between each pair of the two collection, here the collection is the variable coordinates and also coordinates. ‘euclidean’ is to declare which distance metric to use. Line 68 is to show the distanceMatrix and line 70 71, is the visualization of the distance matrix using heatmap.

```
72. def create_chromosome(_names):
73.     chromosome = copy.deepcopy(_names)
74.     np.random.shuffle(chromosome)
75.     return chromosome
```

Line 72 is to define the function create chromosome, with the parameter names. Line 73 has the function to create a new variable called chromosome, that contains the copy of names, and at line 74 the content of chromosome will be shuffled. Line 75 is to return the value chromosome.

```
76. def evalChromosome(_distanceMatrix, _chromosome):
77.     distance = 0
78.
79.     for i in range(len(_chromosome) - 1):
80.         _i = _chromosome[i]
81.         _j = _chromosome[i+1]
82.         distance += _distanceMatrix[_i][_j]
83.
84.     distance += distanceMatrix[_chromosome[-1], _chromosome[0]]
85.     return distance,
```

Here, line 76 is to define another function which is the chromosome evaluation, or in here called evalChromosome, the parameter that will be used is the distance matrix and also the chromosome, line 77 is to declare a new variable called distance, while 79 is to start a loop in the range of chromosome -1 length. In line 82, distance will be added with the distance matrix that contains the value of _i and _j. Line 84 is adding another distance with the distance matrix but with chromosome[-1] and also chromosome[0]. Line 85 is to return the distance value.

```
86. populationNumber = 500
87. maxGeneration= 15000
88. crossoverprob = .6
89. mutationprob = .4
```

Line 86 has the function to determine how many populations will be generated, while line 87 will determine the maximum number of generations. Crossover probability and also mutation probability will also be declared in line 88 and 89.

```
90. tool.register('attribute', random.random)
91. tool.register('index',create_chromosome,names)
92. tool.register('indi',      tools.initIterate,      creator.Individual,
   tool.index)
93. tool.register('popul', tools.initRepeat, list, tool.indi)
94. tool.register('eval', evalChromosome, distanceMatrix)
95. tool.register('select', tools.selTournament)
96. tool.register('mate', tools.cxPartialyMatched)
97. tool.register('mutate', tools.mutShuffleIndexes)
98.
99.
```

```
100.popul = tool.popul(n=populationNumber)
101.fitSet = list(tool.map(tool.eval, popul))
102.print(min(fitSet))
103.for ind, fit in zip(popul, fitSet):
104.    ind.fitness.values = fit
```

Line 90-97 is the process of registering all of the information into the toolbox, line 100 has the function to declare variable population, here population will use tool.popul with the parameter population number that has been declared before. Line 101, is where the population is evaluated, line 101 is the process of mapping the evaluation function to every population, and the next step is to assign the respective fitness.

```
105.for gen in range(0, maxGeneration):
106.
107.    if (gen % 10 == 0):
108.        print(f'Generation: {gen}' )
109.        print(f'Fitness: {fitness:}' )
110.
111.    child = tool.select(popul, len(popul), tournsize=3)
112.    child = list(map(tool.clone, offspring))
```

Line 105 is to start another loop that stop when the maximum number of generation has been reached. Line 107 will conduct another ifelse situation, when the generation is modulo with 50 equals to 0 then the conditions is true, if true then in line 108 the number of generation and the fitness score will be printed. Line 110-111 is the process of selecting the next generations, offspring will be generated, this value contains list of the exact copy from the population.

```
113.for o1, o2 in zip(child[0::2], child[1::2]):
114.        if np.random.random() < crossoverprob:
115.            tool.crossover(o1, o2)
116.            del o1.fitness.values
117.            del o2.fitness.values
118.for chromosome in child:
119.    if np.random.random() < mutationprob:
120.       tool.mutate(chromosome, indpb=0.01)
121.       del chromosome.fitness.values
```

This code is the process of crossover, line 112 is the looping where child1 and child2 in the range of merged offspring. Line 113 will be another ifelse condition where if the random value is lower than the crossover probability then the o1 and o2 will be deleted (line 115-116). Line 114 is where the child is crossovered, crossover method that will be used is the partially matched, as written in line 96. Line 117 is another loop for each chromosome that's available in the offspring and continue in the if else situation, if the random value is higher than the mutation probability, there will be a tool.mutate from line 97 the mutation process that will be used in this is the shuffle index with independent probability for each attribute to be exchanged is 0.01 or 1%. Line 120 is to delete the chromosome fitness value.

```
122.invalid = [ind for ind in child if not ind.fitness.valid]
123.    fitSet = map(tool.eval, invalid)
124.    for ind, fit in zip(invalid, fitnSet):
125.        ind.fitness.values = fit
```

Line 121-124 has the function to map the offspring from the fitness that's marked invalid in the variable invalid_ind.

```
126.    population[:] = offspring
127.
128.    currentSolution= tools.selBest(population, 2)[0]
129.    currentFitness = currentSolution.fitness.values[0]
130.
131.    if currentFitness<best_fit:
132.        solution = currentSolution
133.        fitness = currentFitness
134.
135.    fitness_list.append(fitness)
136.    solution_list.append(solution)
```

Line 125 is to replace the old population with the offspring. Then line 127 is to select the best population, and line 128 is to find the best fitness available. Line 130 explains if the current best fitness is lower than fitness then the best solution and best fitness will be replaced. Line 134-135 is where best fitness and best solution combined with the rest of best fitness/solution list.

Results

Result with 40 Nodes

Result of genetic algorithm when given 40 nodes, genetic algorithm is able to optimize the waste transport route.

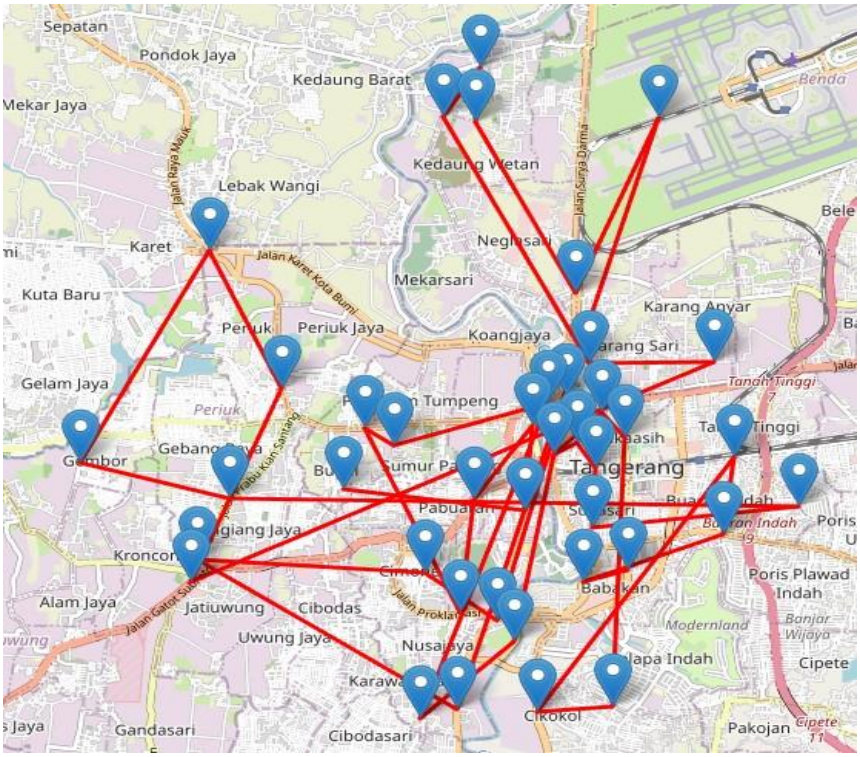

Figure 7. 40 Nodes Initial Route

Figure 5.1 shows the original route from the data, where the route is determined by the number of index and nothing more.

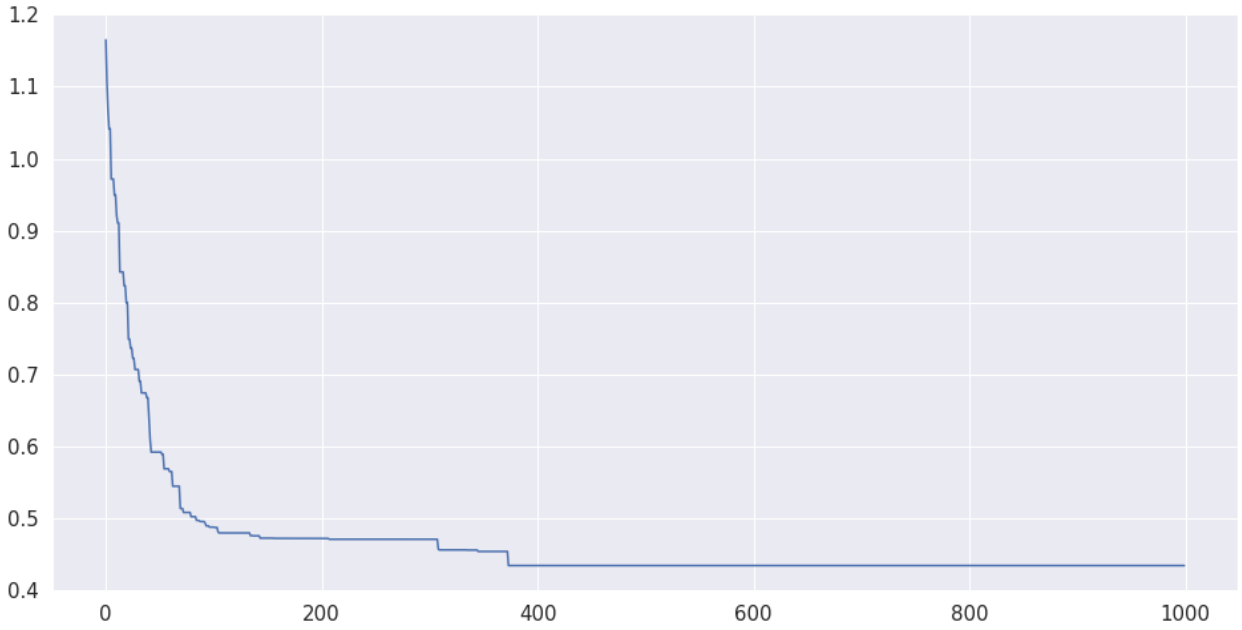

Figure 8. 40 Nodes Fitness Graph

Figure 5.2 shows the process of computing the fitness of genetic algorithm, since the maximum number of generations is 1000, genetic algorithm will stop when generations has reached 1000, the chart shows that the fitness value is determined at about 300-400 generations, with the value less than 0.5.

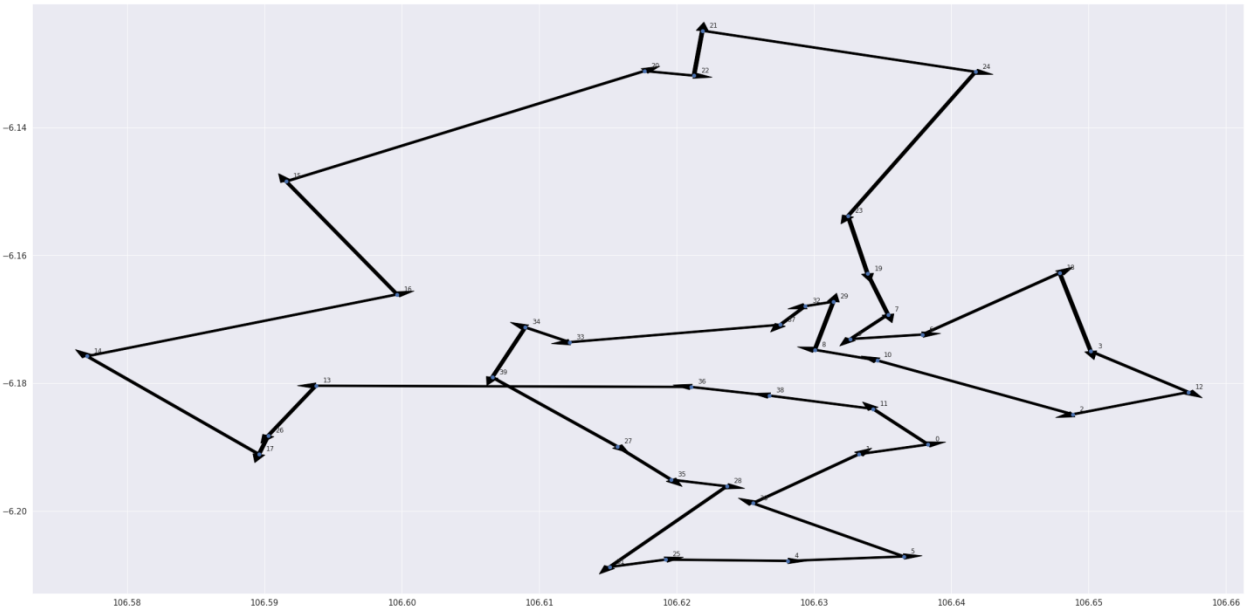

Figure 9. 40 Nodes Genetic Algorithm Route

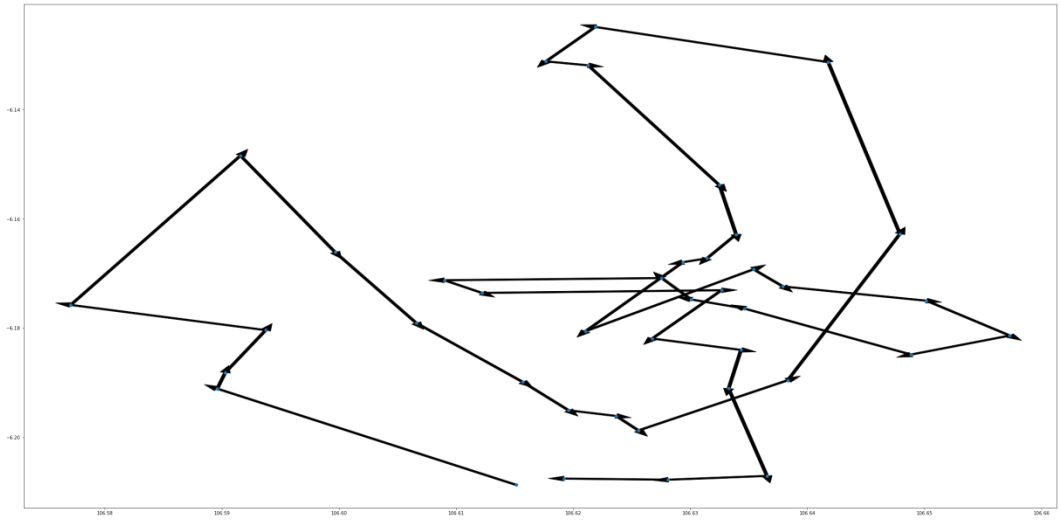

Figure 10. 40 Nodes Brute Force Route

Figure 5.3 and 5.4 shows the route generated by genetic algorithm and also brute force, by the graph the route produced by genetic algorithm shown to be more neat.

Table 7. Before and After Genetic Algorithm Comparison

	Original	**Genetic Algorithm**	**Brute Force**
Result [0]	0.5631612106884862	1.7083730491896456	2.8932641702388286
Route Length	0.5631612106884862	1.7083730491896456	2.8932641702388286
Result [1]	1.7437025959569896	0.3658067586384525	0.12045321306959018
Route Length	2.3068638066454756	2.074179807828098	3.013717383308419
…	…	…	…
Result [38]	2.243642201142117	0.22837772829356223	0.9901011031766647
Route Length	62.16260620824876	35.740669887802426	38.31329870232137
Result [39]	3.4251133284295787	0.22227086178143585	0.45538121382352215
Route Length	65.58771953667834	35.962940749583865	38.76867991614489

Result with 80 Nodes

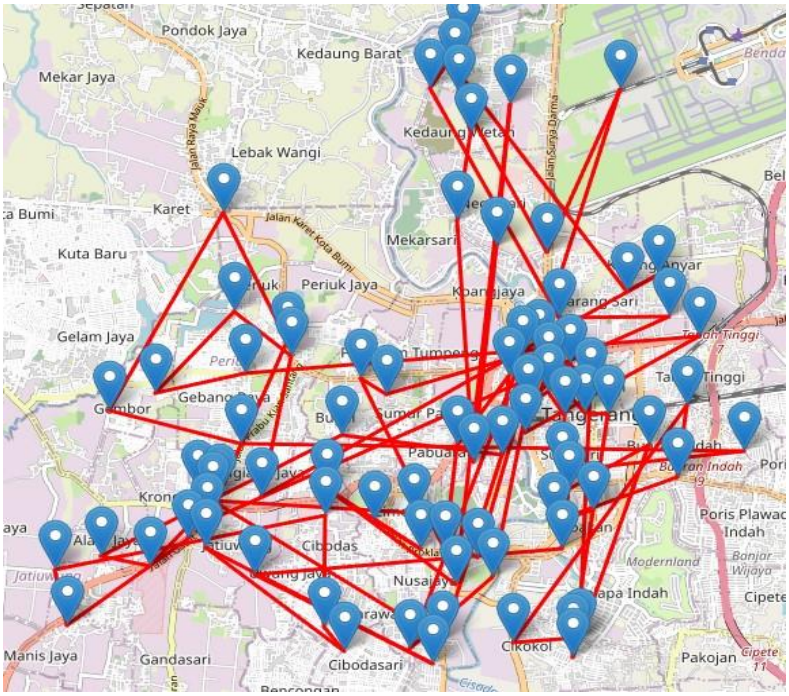

Figure 11. 80 Nodes Initial Route

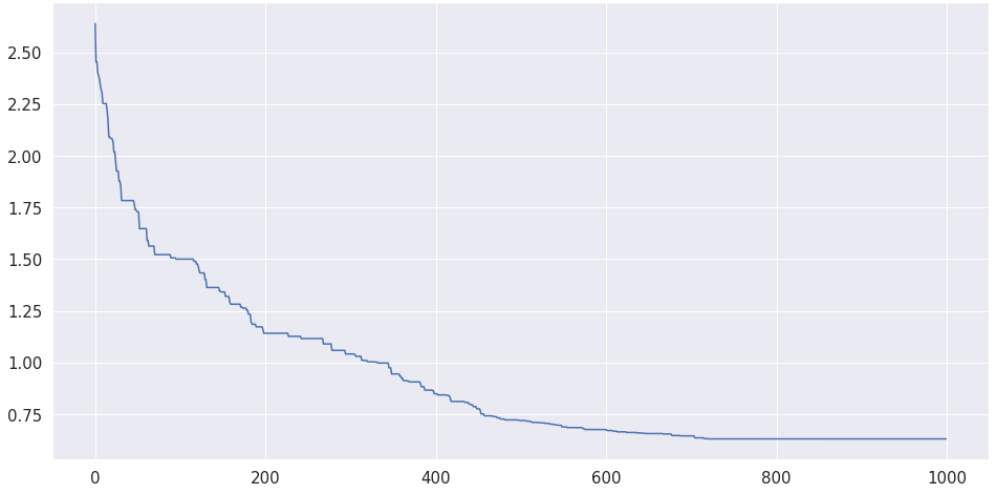

Figure 12. 80 Nodes Fitness G

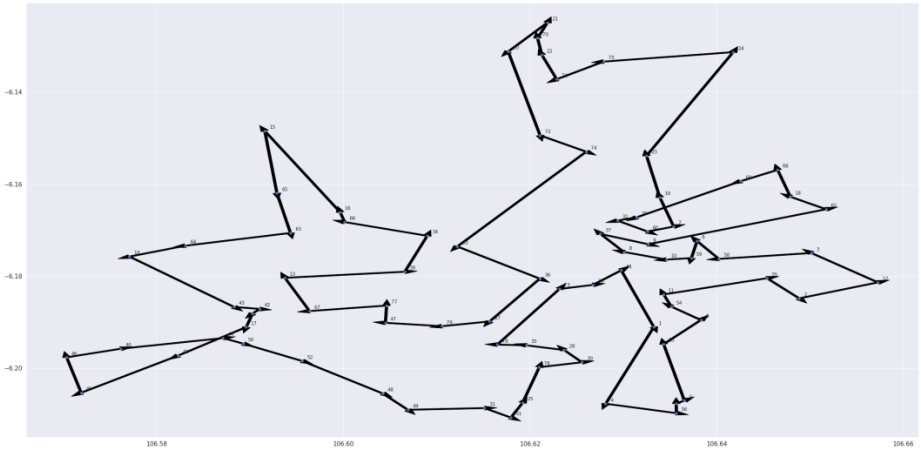

Figure 13. 80 Nodes Genetic Algorithm Route

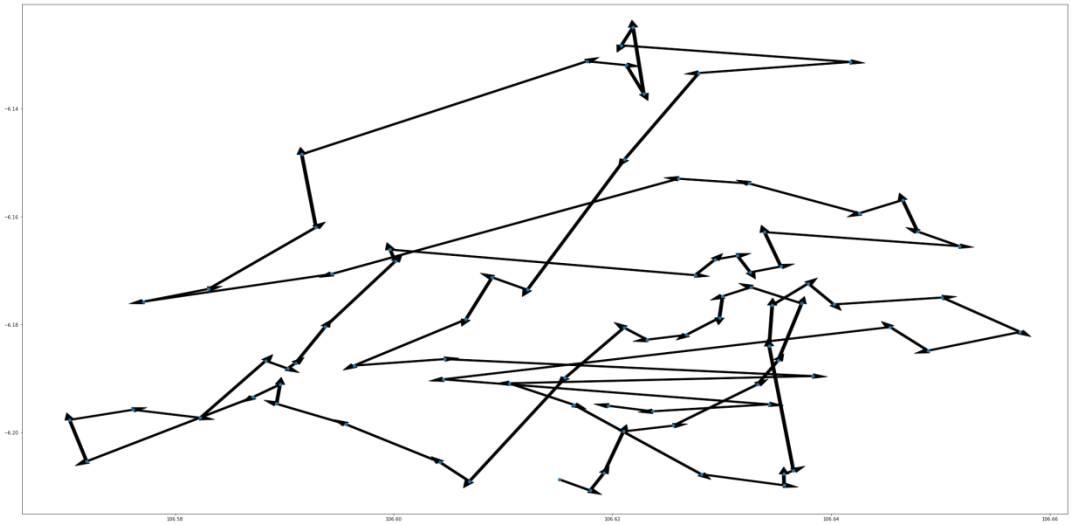

Figure 14. 80 Nodes Brute Force Route

Table 8. Before and After Genetic Algorithm Comparison

	Original	Genetic Algorithm	Brute Force
Result [0]	0.5631612106884862	0.24009883037246396	0.3211068157342795
Route Length	0.5631612106884862	0.24009883037246396	0.3211068157342795
Result [1]	1.7437025959569896	0.44624071854333186	0.17301471897737694
Route Length	2.3068638066454756	0.6863395489157958	0.49412153471165643
…	…	…	…
Result [78]	0.6722039905553483	2.133363596132943	0.45230817911995236
Route Length	123.14582049540456	47.36012963637782	69.13986255574876
Result [79]	2.540565999548715	0.4341426294850353	0.6602499972833791
Route Length	125.68638649495328	47.79427226586286	69.80011255303214

Result with 120 Nodes

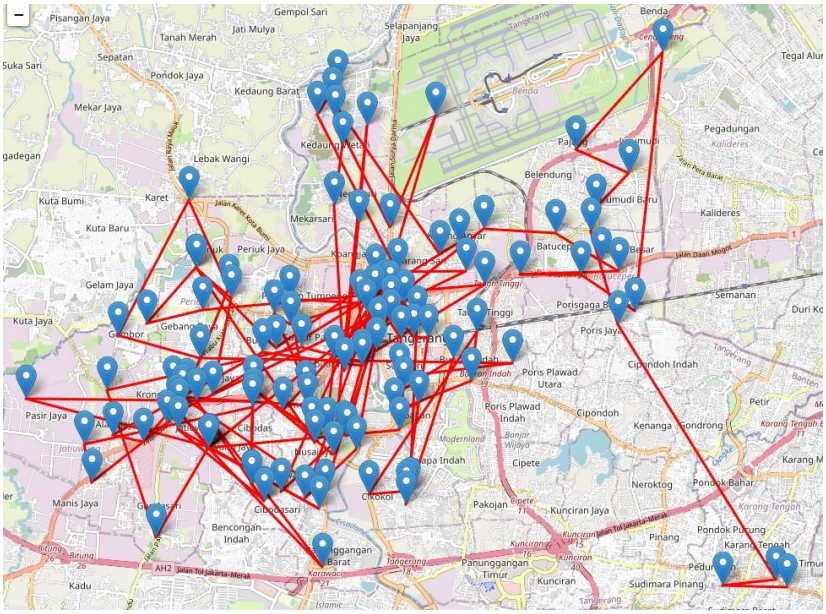

Figure 15. 120 Nodes Initial Route

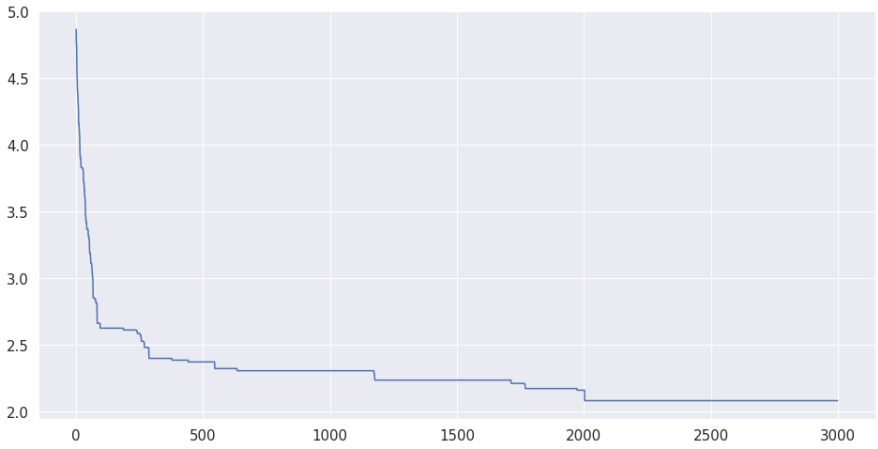

Figure 16. 120 Nodes Fitness Graph

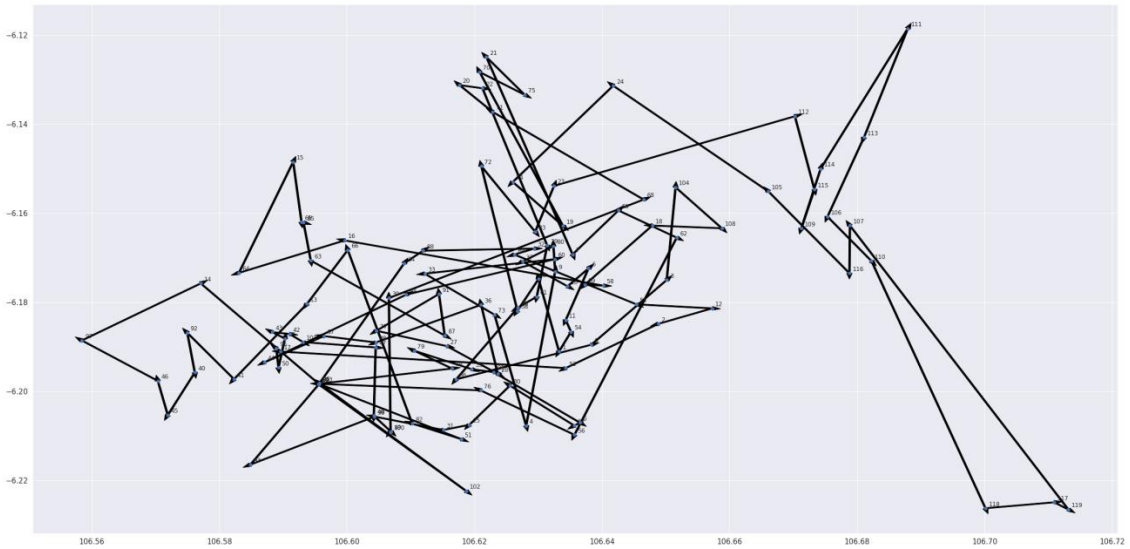

Figure 17. 120 Nodes Genetic Algorithm Route

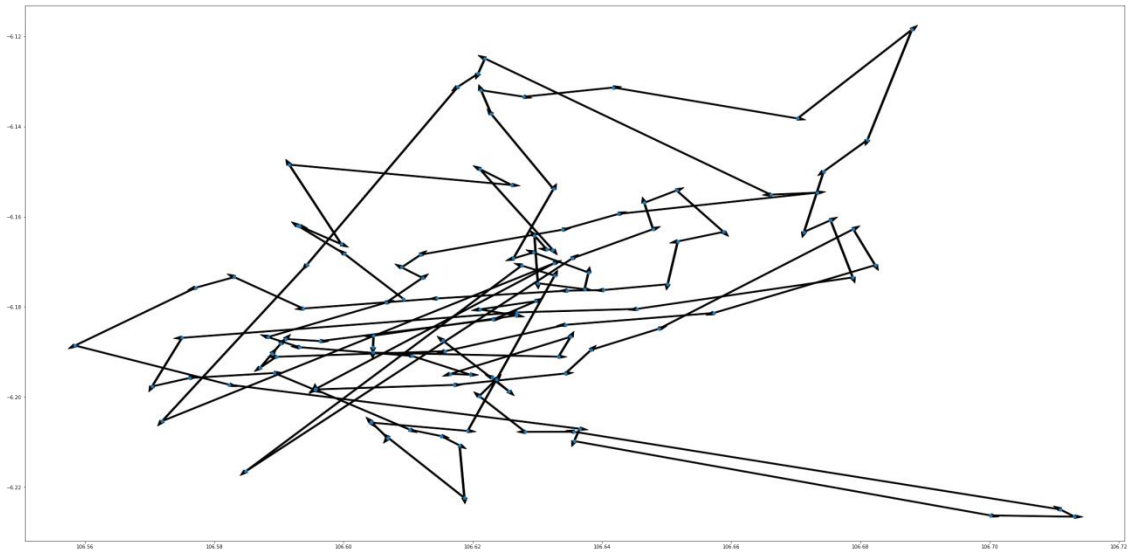

Figure 18. 120 Nodes Brute Force Route

Table 9. Before and After Genetic Algorithm Comparison

	Original	**Genetic Algorithm**	**Brute Force**
Result [0]	0.5631612106884862	1.5923794282150716	1.0664342437622145
Route Length	0.5631612106884862	1.5923794282150716	1.0664342437622145
Result [1]	1.7437025959569896	0.40893632177290185	0.8308951431785069
Route Length	2.3068638066454756	2.0013157499879735	1.8973293869407213
…	…	…	…
Result [118]	1.4462307207016623	0.6088026306313461	1.1920458970030459
Route Length	196.5610145810444	149.07580418904732	176.35224570184624
Result [119]	1.0268353078008376	0.40140381468802305	0.7334800138807783

Route Length	197.58784988884523	149.47720800373534	177.08572571572702

Result with 160 Nodes

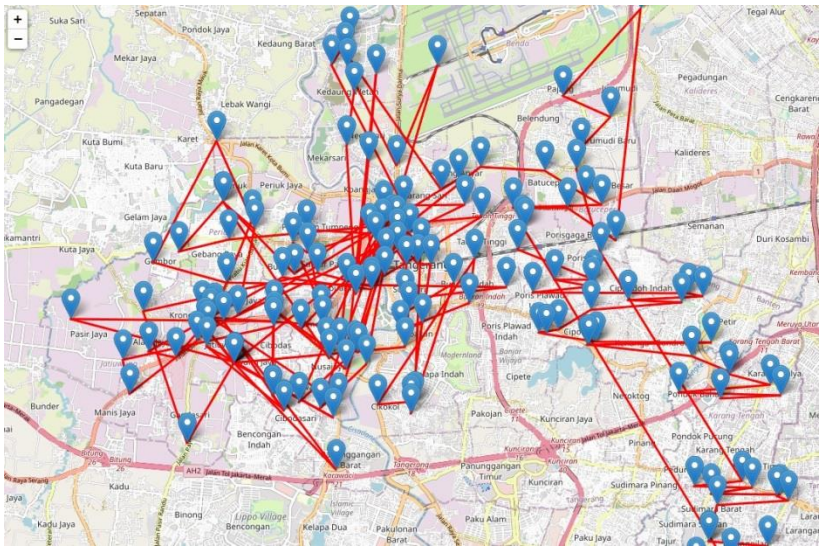

Figure 19. 160 Nodes Initial Route

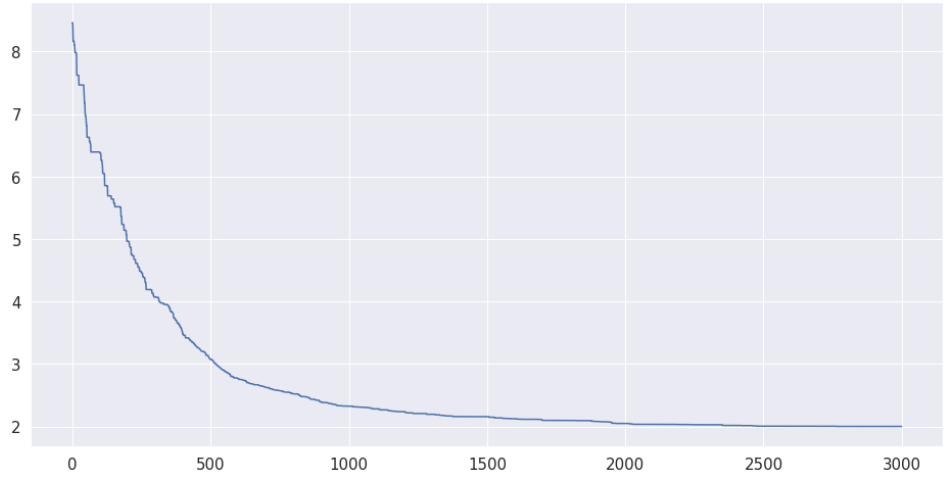

Figure 20. 160 Nodes Fitness Graph

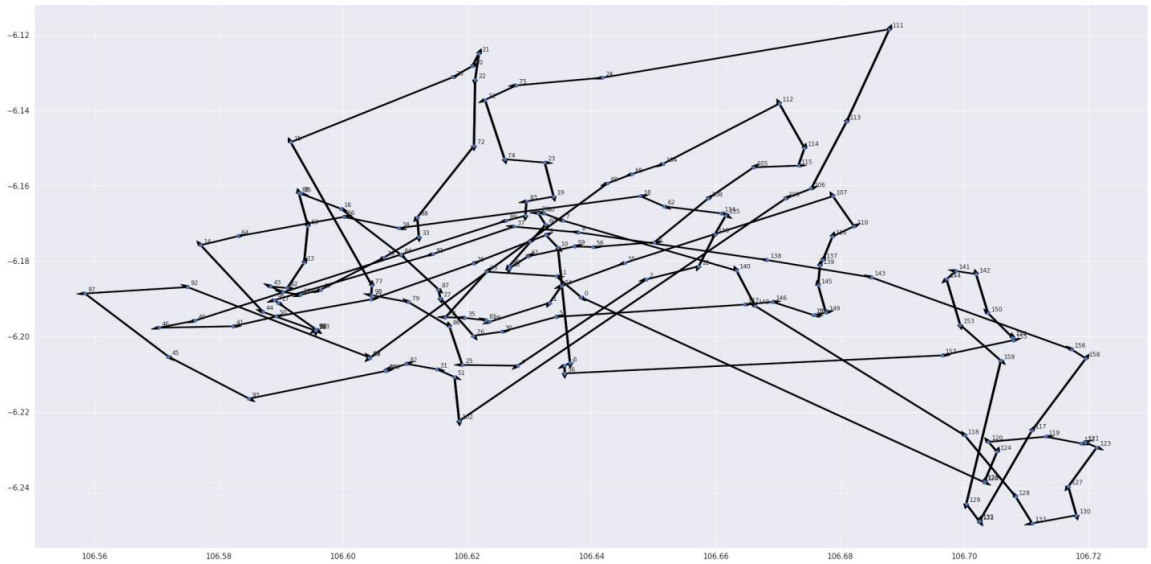

Figure 21. 160 Nodes Genetic Algorithm Route

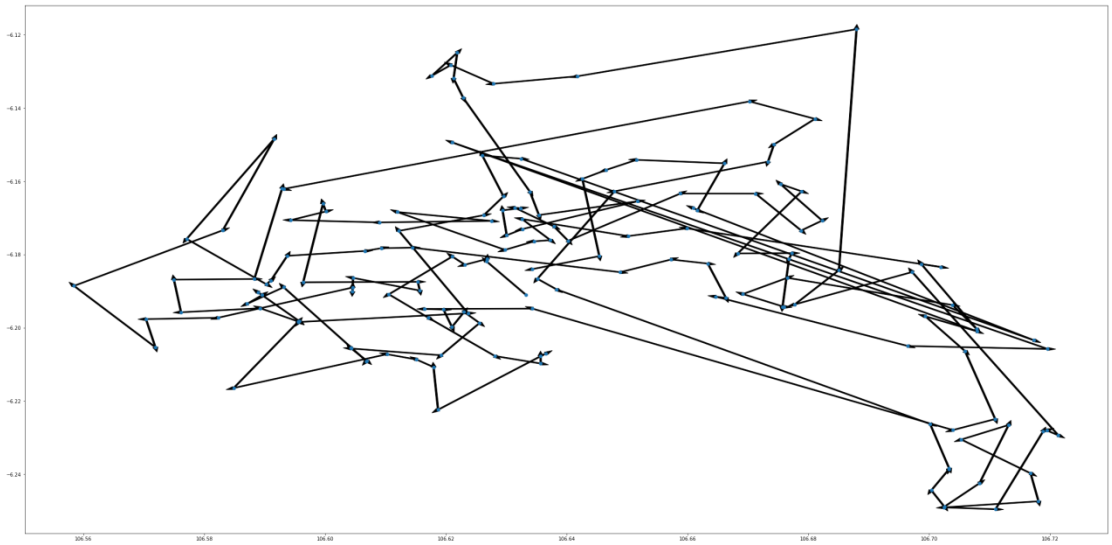

Figure 22. 160 Nodes Brute Force Route

Table 10. Before and After Genetic Algorithm Comparison

	Original	**Genetic Algorithm**	**Brute Force**
Result [0]	0.5631612106884862	0.02903115341224859	0.7813824906675501
Route Length	0.5631612106884862	0.02903115341224859	0.7813824906675501
Result [1]	1.7437025959569896	0.19775520216635067	0.0225142204910293
Route Length	2.3068638066454756	0.22678635557859927	0.8038967111585793
…	…	…	…
Result [158]	1.5214820293907358	0.5023232733131163	1.2360518380234788
Route Length	251.36570578287692	161.49328537426783	210.25904693025487

Result [159]	7.536517853946805	0.7367009841523663	0.2537507246858879
Route Length	258.9022236368237	162.2299863584202	210.51279765494075

CONCLUSION

From the experiment that has been conducted, a conclusion can be taken, from the result it has been proven that genetic algorithm is able to optimize the waste transportation route, and compared with another algorithm which is a Brute Force algorithm, the genetic algorithm has proven to be superior. That being said, there is still a few downside to genetic algorithm, which are; genetic algorithm needs a longer time when computing more data, for example 160 data with 400 populations takes about 20-30 minutes to compute, genetic algorithm also needs a specific population number to produce the most optimum route, and since there is no pattern in the result, a lot of time is consumed to do trial and error.

For the next research, it might be better to try out other optimization algorithm, or create a hybrid genetic algorithm that can produce an even better solution with lower time consumption.

REFERENCES

[1] Sydulu Maheswarapu; Mithun M. Bhaskar M. Bhaskar, "A Hybrid Genetic Algorithm Approach for Optimal Power Flow," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, no. Vol 9, No 2: August 2011, pp. 211–216, 2011.

[2] D. D. F. Rochman Rendiyatna, "PENJADWALAN 20 JOB 8 MESIN DENGAN METODE GENETIC ALGORITHM (GA)," *SPEKTRUM INDUSTRI*, no. Vol 11, No 2: Oktober 2013, 2013, [Online]. Available: http://journal.uad.ac.id/index.php/Spektrum/article/view/1733

[3] Aprilia Nur Fauziyah; Wayan Firdaus Mahmudy, "Hybrid Genetic Algorithm for Optimization of Food Composition on Hypertensive Patient," *International Journal of Electrical and Computer Engineering (IJECE)*, no. Vol 8, No 6: December 2018, pp. 4673–4683, 2018.

[4] Nor Shahida Mohamad Yusop; Marshima Mohd Rosli; Aini Sofea Fazuly, "Design of meal intake prediction for gestational diabetes mellitus using genetic algorithm," *IAES International Journal of Artificial Intelligence (IJ-AI)*, no. Vol 9, No 4: December 2020, pp. 591–599, 2020.

[5] E. P. Armandi Annie; Linarti, Utaminingsih, "OPTIMASI RUTE PENGANGKUTAN SAMPAH KOTA YOGYAKARTA MENGGUNAKAN HYBRID GENETIC ALGORITHM," *Jurnal Ilmiah Teknik Industri*, no. Vol. 18, No. 2, Desember 2019, pp. 236–244, 2019.

[6] Josephin Sundah, "SCHEDULING AND POWER DISTRIBUTION SYSTEM ON DIESEL POWER PLANT IN NORTH SULAWESI USING GENETIC ALGORITHM," *Jurnal Sistem Informasi*, no. ##issue.vol## 9, ##issue.no## 2 (2013): Jurnal Sistem Informasi, pp. 92–94, 2013.

[7] R. D. R. Puspitasari Dian Eka; Fauzi, Mochammad Ali, "Optimasi Susunan Gizi Makanan Bagi Pasien Rawat Jalan Penyakit Jantung Menggunakan Real Coded Genetic Algorithm (RCGA)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, no. Vol 2 No 1 (2018), pp. 44–52, 2018.

[8] M.-D. Yang and T.-C. Su, “An optimization model of sewage rehabilitation,” *Journal of the Chinese Institute of Engineers*, vol. 30, no. 4, pp. 651–659, 2007.
[9] K. Vairavamoorthy and M. Ali, “Optimal design of water distribution systems using genetic algorithms,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 15, no. 5, pp. 374–382, 2000.
[10] R. C. Li Xin; Liu, Hongxia, “The Optimization of Finishing Train Based on Improved Genetic Algorithm,” *Indonesian Journal of Electrical Engineering and Computer Science*, no. Vol 12, No 5: May 2014, pp. 3555–3559, 2014.