# Traveling Salesman Problem Multi-destination Route Recommendation System Using Genetic Algorithm and Google Maps API

Catharina Adinda Mega Cahyani
School of Information Technology
Universitas Ciputra Surabaya
Surabaya, East Java
Indonesia

Trianggoro Wiradinata*
School of Information Technology
Universitas Ciputra Surabaya
Surabaya, East Java
Indonesia

## ABSTRACT

Google Maps does not provide route recommendations if users want to find the shortest route from multiple destinations or stop destinations, or more than two destinations. Departing from the shortcomings of Google Maps which cannot sort the sequence of multi-destination routes with the shortest distance, the researcher created an innovation with a genetic algorithm in solving the problem of the Traveling Salesman Problem category. The processes in the genetic algorithm of solving the Traveling Salesman Problem include data collection from primary document sources, ETL implementation, genetic algorithm implementation, and genetic algorithm testing with comparison algorithms. The data to be used in this study is primary data from the day tour package "Banyuwangi City Tour" from PT. LINTASNUSA TOURISM PRIMARY. This research produces recommendations for destination routes with the shortest real-time travel distance with short computational time. The genetic algorithm that has been programmed will be compared with other Traveling Salesman Problem solving algorithms, namely Nearest Neighbor and Brute Force. Based on the results of testing with primary data, the genetic algorithm is proven to be able to solve the Traveling Salesman Problem with the shortest average distance and the same as the solution of the Brute Force algorithm, which is 42.759 kilometres. The genetic algorithm also successfully recommended destination routes with shorter real-time travel distances or more optimal solutions compared to the Nearest Neighbor algorithm, but the genetic algorithm took 0.9 seconds slower computational time than the Nearest Neighbor algorithm.

## General Terms

Genetic algorithm, Nearest Neighbour, and Brute Force.

## Keywords

Genetic Algorithm, Traveling Salesman Problem, Google Maps API, and Multi-destination Routes.

## 1. INTRODUCTION

Google Maps is a free navigation and mapping application provided by Google. Google Maps ranks first in popular navigation apps with a percentage of 67% of 511 respondents where respondents are smartphone users who use at least three different apps per day [1]. Based on the same survey, 87% of respondents use navigation apps to show directions when driving.

On some occasions, the destination that users want to reach while driving is more than one or two. This is one of the shortcomings of Google Maps, where Google Maps has not provided route recommendations if users want to find the shortest route from multi-destinations or more than two stops or destinations. This study is important because currently there are no tools for tourism actors in Banyuwangi so in determining the travel route there is no guidance and is prone to unoptimized travel routes which can have an impact on losses for both tour entrepreneurs, tourists, and business activities related to tourism.

Google Maps users can only add additional destinations without a sequence of destinations with the shortest estimated distance. To get a sequence of destination routes with the shortest distance on Google Maps, users must drag and drop and make as many route combinations as $(n-1)!$ or n less one and then factor. The number of destinations (n) is reduced by one because the route starts from the departure point and returns to the departure point. Simply put, to get the best route, users must try all possible destinations of the route manually until the route with the shortest distance and fastest travel time is found. This is a weakness in using the Google Maps application because, of course, it will take a lot of time until the route with the shortest distance and the fastest time. Therefore, a recommendation for determining the route of many destinations with the shortest distance is needed. Departing from the shortcomings of Google Maps which cannot sort the order of multi-destination routes with the shortest distance, an innovation was created with a genetic algorithm in solving the Traveling Salesman Problem category. By solving the Traveling Salesman Problem with a genetic algorithm and utilizing Google Maps API to access the distance between destinations in real-time, the best individual will be generated in the form of a route sequence of destinations with the shortest distance.

## 2. LITERATURE REVIEW

### 2.1 Previous Studies

There have been several previous studies that address the topic of solving the Traveling Salesman Problem using genetic algorithms. One of them is a study [2] that used a genetic algorithm to determine the optimal route among four anonymous cities (A, B, C, and D). The route starts from city A and will also return to city A. The distances between cities are determined with default values. The stopping criterion for the genetic algorithm to generate new generations (iterations) is if the obtained fitness value is not better or equal to the fitness value of the previous generation. In this study, that criterion will be adopted, but will be developed with a genetic algorithm for solving the Traveling Salesman Problem with real-time distances from the Google Maps API for real destinations. Meanwhile, a related study on genetic algorithms to solve the Traveling Salesman Problem has been conducted for determining the distribution routes of PT. Pos Indonesia in Bandar Lampung [3]. The distances used to calculate fitness are computed using the Euclidean distance between the longitude and latitude of each post office address, then multiplied by the decimal degree value. The parameters used are as follows: chromosome length of 16, population size of 30, crossover rate of 0.95, mutation rate of 0.01, and number of generations of 100. The limitation of this study is that the calculation of distances between post offices does not use actual travel distances. However, the genetic algorithm still proves to provide an optimal solution with the shortest distance of 66.52239581 km among the 16 post office locations. In this study, real-time distances from the Google Maps API will be used, making it a more realistic solution.

There is international research that also discusses genetic algorithms for solving the Traveling Salesman Problem using genetic algorithms [4]. This study proves that genetic algorithms outperform the Discrete Fruit Fly Optimization and Simulated Annealing algorithms in achieving the best results in two out of three Traveling Salesman Problem cases. The limitation of this study is that the testing of the genetic algorithm is limited to distance matrices in three samples Traveling Salesman Problem datasets and has not been applied to real-world problems. In this study, real-time distances from the Google Maps API will be used, making it a more realistic solution.

Another international research [5] found that genetic algorithms are more efficient compared to a similar algorithm, Particle Swarm Optimization, as genetic algorithms process only a few particles and their operations involve only particle exchanges. The weakness of this research is the implementation of the genetic algorithm that iterates the movement of particles without storing the global best value. Therefore, this study will store the overall best value throughout the iterations or generation generations in the genetic algorithm.

Based on these three previous studies, the application of genetic algorithms is chosen to solve the Traveling Salesman Problem.

## *2.2* Terminology

### *2.2.1 Genetic Algorithm*

A genetic algorithm is an algorithm based on population for optimizing problems that require a large and complex search space. It is through the population that genetic algorithms can find solutions beyond the scope of local optima. When a new generation is created through crossover and mutation processes, chromosomes are evaluated based on a fitness function [6]. The following is the sequence of processes in a genetic algorithm [6]:

1. Initialization: forming a set of individuals by randomly arranging chromosome genes in a specific order. These chromosomes represent the solutions to be found. In this study, the chromosome representation used is permutation representation.
2. Reproduction: the process of producing offspring from individuals in the population. The recombination operators used in the reproduction process are crossover and mutation.
3. Evaluation: Calculate the fitness value of each chromosome. The higher the fitness value, the more suitable the chromosome is as a candidate for the optimal solution.
4. Selection: selecting individuals from the population and offspring to be retained for the next generation. Probabilistic functions are used to select individuals to be retained. Individuals with higher fitness values have a greater chance of being selected. This probabilistic function is known as roulette wheel selection.

Here are some important definitions of genetic algorithms (Belluano in [7]) [8]:

1. Genotype: the representation of the values in a chromosome. Chromosome is the basic unit of genotype. Genotypes can be binary, float, character, combinatorial, and so on.
2. Allele: the representation of a value in a genotype.
3. Chromosome: a collection of genotypes that form specific values.
4. Locus: the position of a gene in a specific chromosome.
5. Individual: the representation of one of the expected optimal solutions. An individual consists of a chromosome with a certain length.
6. Population: a group of individuals processed in a specific cycle.
7. Generation: a unit to represent one cycle of evolution or iteration in genetic algorithms.

### *2.2.2 Crossover*

The crossover process involves inheriting some genes from the parent individuals in the same direction and exchanging other genes. In this stage, the crossover rate (pc) needs to be determined. This value represents the ratio of offspring generated through the crossover to the population size, resulting in offspring equal to pc multiplied by the population size. The crossover method used in this study is the one-cut-point method, randomly selecting a cutting point and performing crossover on the right side of each parent to generate offspring [9][8].

### *2.2.3 Mutation*

In this study, the type of mutation used is reciprocal exchange mutation. This method is the simplest mutation method. The reciprocal exchange mutation works by randomly selecting two positions (exchange points / XP) and swapping the values at those positions [10].

### *2.2.4 Alternative: Nearest Neighbour Algorithm*

The Nearest Neighbour Algorithm is one of the algorithms that apply the Greedy algorithm principle to solve the TSP. The Nearest Neighbour Algorithm chooses the best option based on the most recent data without considering the overall data. The Nearest Neighbour Algorithm is widely used in various fields of science and technology [11] [12]. In solving the Traveling Salesman Problem, the Nearest Neighbour Algorithm always visits the nearest point or destination. The technique is to choose which point or destination to visit first, and as long as there are points that haven't been visited, visit the nearest point that has not appeared in the route, then return to the starting point [13].

### *2.2.5 Alternative: Brute Force Algorithm*

The Brute Force method works by generating all possible routes among all points and then calculating their distances. To obtain the most optimal solution, the route with the shortest distance is chosen. Here are the steps of the Brute Force method for solving the Traveling Salesman Problem [14]:

1. Calculate the total number of destinations or points that need to be visited.
2. Generate and list all possible destination routes.
3. Calculate the distance for each route.
4. Select the route with the shortest distance as the optimal solution.

### 2.2.6 *Traveling Salesman Problem*

The Travelling Salesman Problem (TSP) falls into the category of well-known optimization problems due to the complexity of the computational process and its applications in real life, such as school bus routes and courier vehicle scheduling [15]. The TSP is a combinatorial NP problem. The problem in the Travelling Salesman Problem is described as a salesman and a list of cities. The salesman must visit all the cities, starting from a specific point (e.g., headquarters), and then return to the starting city. The TSP problem uses a permutation representation because it involves finding the optimal sequence of visiting locations, where each location is visited only once, and then finding the sequence with the most optimal value [16]. Mathematically, the TSP can be formulated as a problem of minimizing travel costs as follows [9].

$$Z = \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\right\} \qquad (1.1)$$

with additional arguments as constraint:

$$\sum_{i=1}^{n} x_{ij} = 1, \text{ for } j = 1, 2, 3, \ldots, n-1 \qquad (1.2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1, 2, 3, \ldots, n-1 \qquad (1.3)$$

### 2.2.7 *Permutation Representation*

Permutation representation describes a solution. Each gene in the chromosome is represented by an integer that represents a location, while the locus indicates the order of visitation for each location [9].

### 2.2.8 *Google Maps API*

According to Google Maps, it is a web service that provides various mapping services. Meanwhile, Google Maps API is a JavaScript library that allows the modification of existing maps on Google Maps according to specific needs. Google Maps API helps users maximize the mapping capabilities of Google for specific purposes and benefits [17].

### 2.2.9 *Multidestination Travel Routes*

Multidestination travel routes refer to the Traveling Salesman Probwhichhere allows someone to create a combination of routes to reach one or more desired destinations during the journey [18].

## 3. RESEARCH METHODOLOGY

Broadly, this research will use a multi-destination list from primary data as input. The multi-destination list will serve as a parameter for the distance_matrix method of the Google Maps API, with the API key filled in, to extract a list of road distances between destinations. From this list, it will be transformed into a Traveling Salesman Problem. To find the optimal solution to the Traveling Salesman Problem, a genetic algorithm will be implemented in a Phon programming language. The genetic algorithm program includes population formation, parent selection, mutation, crossover, and fitness value calculation. The output of the program will be a recommendation for the sequence of routes from the entered destinations, with the shortest road distances. The visualization of the analytical method can be seen in the following Figure 1.

In this research, the data comes from primary document sources, namely the "Banyuwangi City Tour" tour package from PT PRATAMA WISATA LINTASNUSA. After data collection, ETL (Extract Transform Load) is implemented on the data. The following Table 1 is a visualization of the ETL diagram in this study.

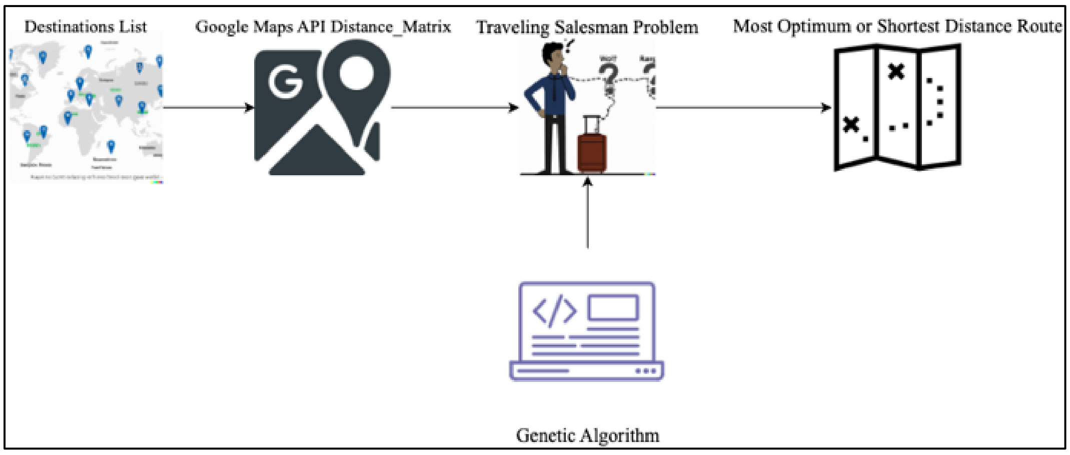
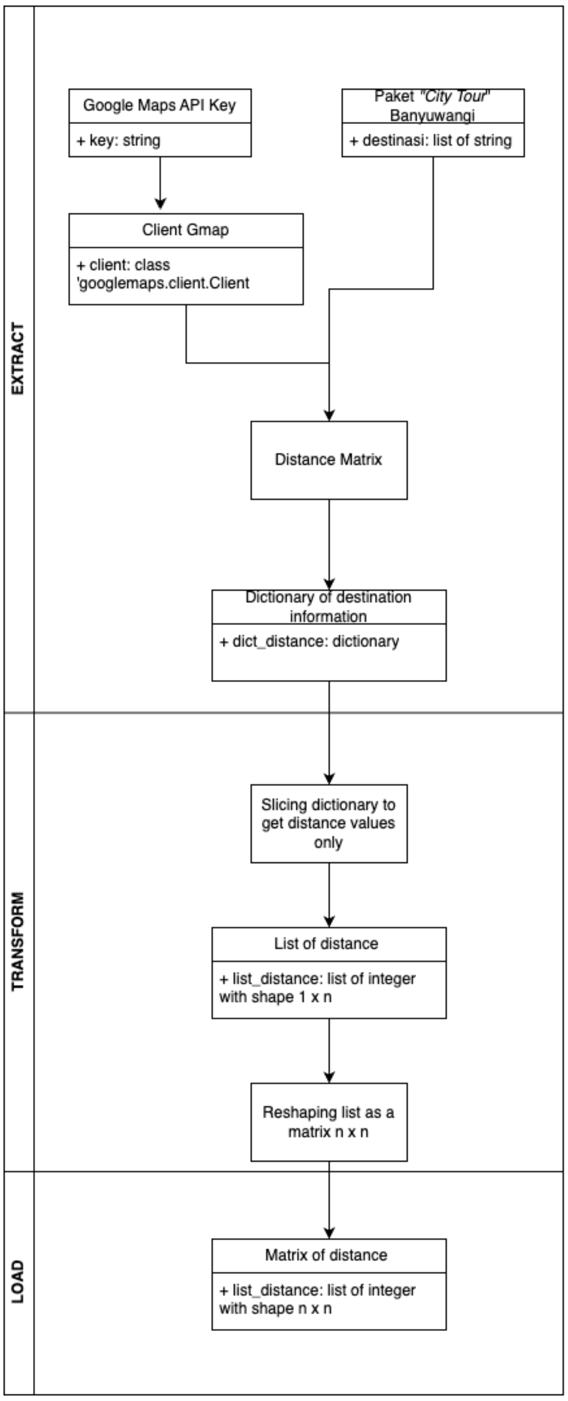

Destinations List
Google Maps API Distance_Matrix
Traveling Salesman Problem
Most Optimum or Shortest Distance Route
Genetic Algorithm

**Figure 1 Research Workflows**

**Table 1 ETL Research Diagram**


EXTRACT
TRANSFORM
LOAD
Google Maps API Key
+ key: string
Paket "City Tour" Banyuwangi
+ destinasi: list of string
Client Gmap
+ client: class 'googlemaps.client.Client
Distance Matrix
Dictionary of destination information
+ dict_distance: dictionary
Slicing dictionary to get distance values only
List of distance
+ list_distance: list of integer with shape 1 x n
Reshaping list as a matrix n x n
Matrix of distance
+ list_distance: list of integer with shape n x n

```
%time
def main():
  starting_point = 1
  progress = []
  if len(location) <= 5:
    population = 20
  elif len(location) <= 7:
    population = 50
  else:
    population = 100
  indiv =
initialization(population,len(location),starting_point)
  fitness_list = fitness(indiv, distance_gmap,
len(location))
  indiv = selection(indiv, fitness_list, population)
  if len(location) <= 6:
    iteration = 20
    cross_rate = 0.2
    mutation_rate = 0.5
  elif len(location) <= 8:
    iteration = 50
    cross_rate = 0.2
    mutation_rate = 0.5
  else:
    iteration = 100
    cross_rate = 0.2
    mutation_rate = 0.7
  i = 0
  fitness_twin = 0
  temp_fitness = 0
  while i < iteration and fitness_twin < 35:
    offspring_cross = crossover_onecut(indiv, cross_rate,
len(location), distance_gmap)
    offspring_mut = mutation(indiv, mutation_rate,
len(location), distance_gmap)
    indiv = np.append(indiv, offspring_cross, axis = 0)
    indiv = np.append(indiv, offspring_mut, axis = 0)
    fitness_list = fitness(indiv, distance_gmap,
len(location))
    indiv = selection(indiv, fitness(indiv, distance_gmap,
len(location)), population)
    indiv = indiv[:population]
    progress.append(100 / fitness_list[0])
    if fitness_list[0] == temp_fitness:
      fitness_twin += 1
    else:
      fitness_twin = 0
    temp_fitness = fitness_list[0]
    i+=1
  indiv_selection = selection(indiv, fitness(indiv,
distance_gmap, len(location)), population)
  fitness_final = fitness(indiv_selection, distance_gmap,
len(location))
  print("Final optimum route: " + str(indiv_selection[0]))
  print("Final optimum distance: ", ((100 /
fitness_final[0]/1000)))
  distance_final = (100 / fitness_final[0]/1000)
  best_indiv = indiv_selection[0].tolist()
  rute = []
  for i in best_indiv:
    rute.append(destinasi[i-1])
  return progress, rute, distance_final
progress, rute, distance_final = main()
```

**Figure 2 Main Function Snippet Code**

After performing the extraction and transformation process, the list of distances between destinations with a dimension of 7 x 7 will be accommodated into a variable that will directly enter the genetic algorithm.

In the implementation of the genetic algorithm, several processes are divided into several functions which are then called in a main function. In this subchapter, each process in the genetic algorithm that has been created in the main function and the function that is called will be described. The main function of the genetic algorithm is shown in Figure 2 above.

To speed up the computation time, the number of individuals in a population, the number of iterations, the crossover rate, and the mutation rate are determined using simple conditioning (if-else statement). This experiment has been proven to improve the efficiency of the genetic algorithm.

After the determination, initialization is performed to generate the initial population to obtain a set of individuals with a certain number in a population. The random method (shuffle) is used to form the initial individuals, n in a population. In accordance with the condition that the route always starts with a starting point, a chromosome arrangement condition is set to always place the gene index representing the starting point at index 0 (zero). The temporary chromosome formed is then appended with the starting point gene at the last index. This also follows the condition that the route must return to the starting point. The result of this function is a set of candidate parent individuals with random chromosome arrangements.

After initialization, evaluation is performed. Evaluation is used to calculate the fitness of each chromosome of the individual. In this evaluation, the 7 x 7 distance matrix, the result of the ETL process, is used to calculate fitness. The fitness formula used is 100 divided by the total distance of the chromosome arrangement of each individual. The function will return a list or array containing fitness values.

Selection functions to retain individuals with the best fitness. Bubble sorting is performed where there will be continuous element exchanges in the fitness array and individuals as many times as the length of the fitness array but reversed or starting from the last index. The selection function will return an array or list of individuals that have been sorted in ascending order. The selection applied in this research is roulette wheel selection.

After selection, the next stage is reproduction. The genetic algorithm operators used in this reproduction stage are crossover and mutation.

The crossover method used is a one-cut-point, where the offspring or child is formed by exchanging a pair of parents that have undergone a previous selection and roulette wheel processes. In the crossover process, a cut point index is determined randomly for crossover, except for the first and last indexes, forming a part of the offspring chromosome genes by slicing the first parent according to the cut point index, then filling in the remaining genes from the second parent that are not the same as the previous first parent genes. The result of this crossover function is an array or list of offspring equal to the number of individuals in a population multiplied by the predetermined crossover rate through previous conditional statements.

In this research, the reciprocal exchange mutation method is used, which works by selecting two positions or indexes randomly, except for the first and last indexes, and swapping the values at those positions. In the mutation stage, only one selected individual is used as a parent and two indexes are determined to swap their element positions. The result of this mutation function is an array or list of offspring equal to the number of individuals in a population multiplied by the predetermined mutation rate through previous conditional statements.

In addition to the implementation of the genetic algorithm, this research also implements the Nearest Neighbour and Brute Force algorithms. The recommended routes from the genetic algorithm will be compared in terms of distance traveled with the Nearest Neighbour algorithm and computation time with the Brute Force algorithm.

The Nearest Neighbour algorithm implementation implements the queue data structure used by following the FIFO (First In First Out) principle. The addition of elements (enqueue) uses the built-in append and insert functions while the removal of elements (dequeue) with the built-in remove function. The result of the Nearest Neighbour algorithm is the destination route, the accumulated distance between destinations in route order, and computation time.

The implementation of the Brute Force algorithm utilizes the permutations module of the itertools library as well. To get all destination route permutations of $(n-1)!$. After obtaining all permutations of destination routes, all distances between destinations from these permutations are calculated, and then take the destination route with the shortest or minimum distance. The result of this Brute Force algorithm is the destination route, the accumulated distance between destinations in the order of the route, and the computation time.

## 4. RESULTS

The testing process was conducted on Tuesday and Thursday. The first test was conducted on Tuesday, March 28, 2023, at 12:23:41 WIB, comparing the route sequence, distance traveled, and computation time between the genetic algorithm, Nearest Neighbour algorithm, and Brute Force algorithm. The second test will perform the same comparisons on Thursday, April 6, 2023, at 11:04:31 WIB. The final test involves comparing the distances between destinations generated by the genetic algorithm with the original destination routes from the "Banyuwangi City Tour" package.

The results of the first test are presented in Table 2. Based on Table 2, it can be concluded that the implementation of the genetic algorithm shows a significantly shorter computation time compared to the Brute Force algorithm. However, when compared to the Nearest Neighbour algorithm, it appears that the Nearest Neighbour algorithm has a faster computation time with a delta of 0.98 seconds compared to the genetic algorithm.

Based on the aspect of distance traveled for the destination route, it can be concluded that the genetic algorithm is superior to the Nearest Neighbour algorithm and is equally superior to the Brute Force algorithm.

The results of the second test are presented in Table 3.

**Table 2 First Trial Results**

| Algorithm | Destination Route Travelling Distance (kilometre) | Computational Time (seconds) | Destination Route Order |
|---|---|---|---|
| Genetic Algorithm | 42,873 | 1,45 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Jagir Waterfall,<br>3. Gandrung Terakota Park,<br>4. Banyuwangi Jopuro Tourism Attraction,<br>5. Marina Boom Beach Banyuwangi,<br>6. Pendopo Sabha Swagata Banyuwangi<br>7. ASTON Banyuwangi Hotel and Conference Center |
| Nearest Neighbour | 44,706 | 0,542 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Pendopo Sabha Swagata Banyuwangi,<br>3. Marina Boom Beach Banyuwangi,<br>4. Jagir Waterfall,<br>5. Banyuwangi Jopuro Tourism Attraction,<br>6. Gandrung Terakota Park,<br>7. ASTON Banyuwangi Hotel and Conference Center |
| Brute Force | 42,873 | 15,4 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Jagir Waterfall,<br>3. Gandrung Terakota Park,<br>4. Banyuwangi Jopuro Tourism Attraction,<br>5. Marina Boom Beach Banyuwangi,<br>6. Pendopo Sabha Swagata Banyuwangi,<br>7. ASTON Banyuwangi Hotel and Conference Center |

**Table 3 Second Trial Results**

| Algorithm | Destination Route Travelling Distance (kilometre) | Computational Time (seconds) | Destination Route Order |
|---|---|---|---|
| Genetic Algorithm | 42,735 | 1,33 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Jagir Waterfall,<br>3. Gandrung Terakota Park,<br>4. Banyuwangi Jopuro Tourism Attraction,<br>5. Marina Boom Beach Banyuwangi,<br>6. Pendopo Sabha Swagata Banyuwangi<br>7. ASTON Banyuwangi Hotel and Conference Center |
| Nearest Neighbour | 44,828 | 0,633 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Pendopo Sabha Swagata Banyuwangi,<br>3. Marina Boom Beach Banyuwangi,<br>4. Jagir Waterfall,<br>5. Banyuwangi Jopuro Tourism Attraction,<br>6. Gandrung Terakota Park,<br>7. ASTON Banyuwangi Hotel and Conference Center |
| Brute Force | 42,735 | 12,5 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Jagir Waterfall,<br>3. Gandrung Terakota Park,<br>4. Banyuwangi Jopuro Tourism Attraction,<br>5. Marina Boom Beach Banyuwangi,<br>6. Pendopo Sabha Swagata Banyuwangi,<br>7. ASTON Banyuwangi Hotel and Conference Center |

Based on the previous two tables, the author summarizes the test results in Table 4 below.

**Table 4 Quantitative Summary Result**

| Algorithm | Destination Route Travelling Distance (kilometre) | Computational Time (seconds) |
|---|---|---|
| Genetic Algorithm | Excellent | Excellent |
| Nearest Neighbour | Less Excellent | Excellent |
| Brute Force | Excellent | Less Excellent |

Based on the summary, the genetic algorithm is the most superior to solving the Traveling Salesman Problem, especially when compared with data obtained from primary sources, namely "Banyuwangi City Tour". The test results of the recommended destination route sequence of the genetic algorithm with the destination route sequence from the primary source are presented in Table 5 of the following comparison.

**Table 5 Primary Data Comparison Trial Result**

| Destination Route Source | Destination Route Travelling Distance (kilometre) | Destination Route Order |
|---|---|---|
| Genetic Algorithm | 42,7 | 1. ASTON Banyuwangi Hotel and Conference Center,<br>2. Jagir Waterfall,<br>3. Gandrung Terakota Park,<br>4. Banyuwangi Jopuro Tourism Attraction,<br>5. Pantai Marina Boom Banyuwangi,<br>6. Pendopo Sabha Swagata Banyuwangi<br>7. ASTON Banyuwangi Hotel and Conference Center |
| "Banyuwangi City Tour" package on YukBanyuwangi website | 66,5 | 1. Hotel (ASTON Banyuwangi Hotel and Conference Center),<br>2. Banyuwangi Jopuro Tourism Attraction,<br>3. Pendopo Sabha Swagata Banyuwangi<br>4. Jagir Waterfall<br>5. Gandrung Terakota Park,<br>6. Pantai Marina Boom Banyuwangi,<br>7. Hotel (ASTON Banyuwangi Hotel and Conference Center). |

## 5. DISCUSSION

Based on the previous test results, it can be summarised as follows.

1. The genetic algorithm managed to get a solution with the shortest distance and the same as the solution of the Brute Force algorithm, which is about 42.7 kilometres. The genetic algorithm also takes a much shorter computation time with a difference of about 11 seconds.
2. The genetic algorithm managed to get a solution with a shorter distance than the solution from the Nearest Neighbour algorithm with a difference of about 1.7 kilometres. However, the computation time of the genetic algorithm is slightly longer by about 0.9 seconds than the Nearest Neighbour algorithm.
3. The Brute Force algorithm and the genetic algorithm produce the most optimal solution while the Nearest Neighbour algorithm takes the shortest computation time.
4. The genetic algorithm in this study uses a population size of 50, number of iterations of 50, crossover rate of 0.2, and mutation rate of 0.5.

In addition, based on the hypothesis and previous test results, the following conclusions can be drawn.

1. The application of genetic algorithms and Google Maps API, has been able to recommend real-time travel routes with the shortest distance for the multidestination Traveling Salesman Problem in real cases.
2. The application of genetic algorithms, based on the tests written in Chapter 5.1, has been able to recommend destination routes with shorter real-time travel distances or more optimal solutions compared to the Nearest Neighbour

algorithm but genetic algorithms take longer computing time than the Nearest Neighbour algorithm.

## 6. CONCLUSION AND FUTURE STUDIES

### 6.1 Conclusion

The genetic algorithm for solving the Traveling Salesman Problem for a multi-destination shortest route recommendation system has been successfully implemented through the following steps:

1. Data collection from primary document sources.
2. Implementation of ETL (Extract Transform Load).
3. Implementation of the genetic algorithm.
4. Testing the genetic algorithm against comparison algorithms.

The constructed genetic algorithm is capable of producing the most optimal solution, where the solution generated is the same as the solution from the Brute Force algorithm but with a faster computation time compared to the Brute Force algorithm. The built genetic algorithm is able to produce a more optimal solution than the Nearest Neighbour algorithm but at a slower pace.

### 6.2 Future Studies

Development suggestions that could be useful for future studies in this research include:

1. Conduct trials with a larger number of destinations to measure how optimal and fast the genetic algorithm that has been built.
2. Adding constraint variables for fitness in the form of destination density or peak hour measures, destination priority weights (for example: the beach must be visited last in order to enjoy the sunset), destination operating hours, and the like.
3. Testing a variety of real Traveling Salesman Problem cases and modifying the genetic algorithm accordingly.
4. Expanding the options of trip types such as not returning to the starting point or having a final destination that must be fulfilled.
5. Accommodating other distance units besides metres/kilometres such as miles.

## 7. REFERENCES

[1] R. Panko, “The Popularity of Google Maps: Trends in Navigation Apps in 2018,” Jul. 10, 2018. https://themanifest.com/app-development/trends-navigation-apps (accessed Feb. 25, 2023).

[2] E. Wulan and N. Apriani, “The Application of Genetic Algorithm in Solving Traveling Salesman Problem,” 2020, doi: 10.4108/eai.11-7-2019.2297522.

[3] S. Rohman, L. Zakaria, A. Asmiati, and A. Nuryaman, “Optimisasi Travelling Salesman Problem dengan Algoritma Genetika pada Kasus Pendistribusian Barang PT. Pos Indonesia di Kota Bandar Lampung,” Jurnal Matematika Integratif, vol. 16, no. 1, p. 61, 2020, doi: 10.24198/jmi.v16.n1.27804.61-73.

[4] S. Sharma and V. Jain, “A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem,” IOP Conf Ser Mater Sci Eng, vol. 1116, no. 1, p. 012194, 2021, doi: 10.1088/1757-899x/1116/1/012194.

[5] P. Mudjihartono, T. Tanprasert, and R. Setthawong, “A Comparative Study of Modified PSO Algorithm and Traditional PSO and GA in Solving University Course Timetable Problem,” 2018.

[6] M. Gen and R. Cheng, Genetic Algorithms and Engineering Optimization (Engineering Design and Automation), 1st ed. Wiley-Interscience, 1999.

[7] C. Pramartha and H. Suputra, “Rekomendasi Rute Perjalanan Wisata Berbasis Web Menggunakan Algoritma Genetika,” Jurnal Ilmu Komputer, vol. 13, pp. 21–27, Apr. 2020, doi: 10.24843/JIK.2020.v13.i01.p03.

[8] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1st ed. Addison-Wesley Professional, 1989. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=8ac0783ba24b71236b695cbdfab2ca67

[9] W. F. Mahmudy, “Algoritma Evolusi,” Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang, pp. 1–101, 2013.

[10] J. Juwairiah, D. Pratama, H. Rustamaji, H. Sofyan, and D. Prasetyo, “Genetic Algorithm for Optimizing Traveling Salesman Problems with Time Windows (TSP-TW),” International Journal of Artificial Intelligence & Robotics (IJAIR), vol. 1, p. 1, Oct. 2019, doi: 10.25139/ijair.v1i1.2024.

[11] S. A. Nene and S. K. Nayar, “A simple algorithm for nearest neighbor search in high dimensions,” IEEE Trans Pattern Anal Mach Intell, vol. 19, no. 9, pp. 989–1003, 1997, doi 10.1109/34.615448.

[12] I. Sutoyo, “Penerapan Algoritma Nearest Neighbour untuk Menyelesaikan Travelling Salesman Problem,” Jurnal Khatulistiwa Informatika, vol. 20, no. 1, pp. 101–106, 2018, doi: 10.31294/p.v20i1.3155.

[13] G. Kizilateş and F. Nuriyeva, “On the Nearest Neighbor Algorithms for the Traveling Salesman Problem,” in Advances in Computational Science, Engineering and Information Technology, D. Nagamalai, A. Kumar, and A. Annamalai, Eds., Heidelberg: Springer International Publishing, 2013, pp. 111–118.

[14] A. Rahman Saiyed, “The Traveling Salesman Problem,” 2012.

[15] R. H. Warren, “Solving the traveling salesman problem on a quantum annealer,” SN Appl Sci, vol. 2, no. 1, p. 75, 2019, doi: 10.1007/s42452-019-1829-x.

[16] S. Juneja, P. Saraswat, K. Singh, J. Sharma, D. Majumdar, and S. Chowdhary, Travelling Salesman Problem Optimization Using Genetic Algorithm. 2019. doi: 10.1109/AICAI.2019.8701246.

[17] F. Mahdia and F. Noviyanto, “211271-Pemanfaatan-Google-Maps-Api-Untuk-Pemban,” vol. 1, pp. 162–171, 2013.

[18] H. Santoso and R. Sanuri, “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, doi: 10.34148/teknika.v8i2.187.

## 8. AUTHOR'S PROFILE

**Catharina Adinda Mega Cahyani** is an Information System fresh graduate, with research interests in Evolutionary Computing, Deep Learning, Artificial Intelligence, and iOS Mobile Development. Catharina is an alumnus of the Apple Developer Academy @ UC and Bangkit Academy 2022, and a Machine Learning Student by Google, GoTo, and Traveloka. Catharina has been recognized as 1st Runner Up in the National AI Innovation Challenge and has been a finalist in several national data competitions. With her expertise in iOS development and passion for cutting-edge technologies, Catharina contributes to the advancement of intelligent mobile applications and technologies.

**Trianggoro Wiradinata** is a researcher at the School of Information Technology at Ciputra University. His main research interests are data science, software engineering, technology adoption, and technology-based entrepreneurship. He was actively involved in many large-scale enterprise software development projects before joined as researcher at Ciputra University, Surabaya, Indonesia. Assoc. Prof. Dr. Wiradinata is currently a. member of Association for Computing Machinery (ACM) and Association for Information Systems Indonesia (AISINDO).