The Comparison of Brute Force, Cheapest-Insertion, and Nearest-Neighbor Heuristics for Determining the Shortest Tour for Visiting Malls in Bandar Lampung

Meli Amelia, Ilma Isyahna Sholeha, Yanda Rico Revangga, Wamiliana *
Department of Mathematics, FMIPA, Universitas Lampung, Indonesia
meliameliathefirst@gmail.com, ilmaisyahna@gmail.com, yandarevangga3@gmail.com,
* wamiliana.1963@fmipa.unila.ac.id

ABSTRACT – A mall is a sizable retail complex made up of numerous stores, as well as restaurants and other commercial spaces, all situated in one huge building or a collection of related or neighboring buildings. In this study we will compare the heuristics which are Brute Force, Cheapest-Insertion, and Nearest-Neighbor heuristics to find the shortest route from one mall to other eight malls in Bandar Lampung exactly once and back to the origin. The problem of finding the shortest route from original location, and go to every location exactly once, and back to original location is known as The Travelling Salesman Problem. Implementing on the data of the locations of those nine malls, the Brute Force Heuristic gives the total length 32,8 km, Cheapest-Insertion Heuristic 33,2 km, and Nearest-Neighbor Heuristic 33,35 km. However, in term of efficiency, the Brute Force is not efficient because we have to search for all possible solutions.

Keywords: Brute Force; Cheapest Insertion; Heuristics; Nearest Neighbor

1. INTRODUCTION

In the modern era, optimizing travel routes has become important in various fields such as logistics, transportation, and city management. Traveling Salesman Problem (TSP) is a combinatorial optimization problem that aims to find the shortest route for a salesman who must visit each city exactly once and return to the point of origin [1]. TSP has wide applications, including in planning trips to visit various locations in cities [2].

Bandar Lampung, as one of the big cities in Indonesia, has some shopping centers (malls) that are spread across various locations. Determining the optimal route to visit all these malls is very important for time and cost efficiency, both for business and recreational purposes. However, finding the optimal route in the TSP context becomes increasingly complex as the number of locations that must be visited increases. The aim of this research is to find the TSP for visiting several malls in Bandar Lampung using three different algorithms: Brute Force, Cheapest Insertion Heuristic (CIH), and Nearest-Neighbor Heuristic (NNH).

The TSP is considered as one of classical problems in network design problem, and it is very popular. That problem was formulated mathematically by an Irish mathematician, Willian Rowan Hamilton. He created the icosian game, a mathematical game, in 1856. That game entails locating a cycle on a dodecahedron called a Hamiltonian cycle, which uses the edges of the dodecahedron to pass through each of its vertices. Hamiltonian cycles get their name from his work on this game. The TSP has the same properties as this game; therefore, the TSP is also well known as one of the Hamiltonian problems. The TSP gained popularity in scientific circles in the United States and Europe throughout the 1950s and 1960s after moves toward addressing it might earn prizes from the RAND Corporation in Santa Monica [3].

TSP is presented using a complete weighted graph $G = (V, E)$ where V is a set of modes that represent locations to be visited and E is a set of edges e_{ij} that connect the nodes i and j in V. The edges represent the roads that connect the locations. Associated with every edge, there is a weight c_{ij}, where the weight is nonnegative. The weight represents distance/cost/time needed to connect between node i and node j.

There are many researchers had investigated the methods for solving TSP. The Brute Force heuristic had been used to solve the TSP problem for example by [4] for goods distribution; [5], and [6]. The Cheapest Insertion Heuristic (CIH) had been investigated by [7-11]; and Nearest Neighbor Heuristic by [12-13], etc.

In the Section 2 we will give the methods about the heuristics and information about the data used; in Section 3 the results and discussion, and the conclusion is given in Section 4.

2. RESEARCH METHODOLOGY

The Brute Force

One of the fundamental algorithms used in search techniques is called brute force. It selects the least expensive option by comparing the results after every possibility has been explored. When brute force is applied to the Traveling Salesman Problem (TSP), the outcome is undoubtedly optimal, but it is less effective when there are more points—for example, if there are 20, we must look for every possible solution! [5]. All efficiency restrictions, however, become less important if the problem is tackled using software or a programming language and involves a limited number of points. The following are the stages involved in using a brute force approach to complete TSP:

(1) Calculate the total number of tours with (n)! where n represents a node (location).
(2) Create a picture or list of all possible tours.
(3) Calculate the distance of each tour
(4) Choose the shortest tour; this is the optimal solution.

The Cheapest Insertion Heuristic (CIH)

The CIH algorithm adds new nodes gradually after making a tour of the smallest cycles with the fewest weights. To find the least embedding value, new node and edge selection are done concurrently. Between the two nodes that make up the chosen edge, the new node is added [11]. CIH procedure is as follow [10]:

(1) Searching process starts from initial node to the nearest node to create a subtour.
(2) Insert a node under consideration to that subtour so that the added node contributes the smallest weight to the new subtour. Suppose that we have tour $v_i – v_j – v_i$, and we want to add edge e_{ik} and include node k in the new subtour, then the value of added weight/distance is $c_{ik} + c_{kj} - c_{ij}$, where c_{ik} is the weight/distance between nodes i and k.
(3) Repeat steps b until all nodes include in the new tour.

Nearest Neighbor Heuristic (NNH)

Similar to the CIH algorithm, the Nearest-Neighbor Heuristic (NNH) algorithm solves the TSP problem but the solution may not be optimal. The NNH algorithm's optimal decision is determined by the available data only, not by taking into account all previous data (Nene and Nayar, 1995). The following are the steps for NNH:

(1) Determine the starting node of the tour which is also the ending node of the tour.
(2) From this starting node, determine other nodes that are connected to it and have not yet been visited.
(3) Choose the smallest weight of the edge connecting the starting node with all the unvisited nodes.
(4) The node on the selected edge becomes the starting node for the next search and then the node is marked as visited.
(5) Repeat steps (2) to (4) until no more unvisited nodes are found.

The Data

There are nine malls under consideration which are: (1) Ciplaz Lampung, (2) Mall Boemi Kedaton, (3) Transmart Bandar Lampung, (4) Ramayana Tanjung Karang, (5) Chandra Pasar Bawah, (6) Simpur Center, (7) Mall Kartini, (8) Central Plaza, and (9) Lampung City Mall. The data of distance among the nine malls is collected via Google Maps, which is accessed on December 16th 2023 from 07.00 P.M to 07.46 P.M. The data collected is based on the car path recommended by Google.

Table 1 The Data of the Distance Among the Nine Malls in Bandar Lampung

Node	Distance (km)								
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1	0	3.9	9.1	6.9	7.3	7.3	8.1	9.0	13
v_2	2.9	0	5.3	3.4	3.8	3.9	4.7	5.6	9.1
v_3	6.1	3.2	0	6.1	6.4	6.5	7.3	8.2	9.9
v_4	6.9	4.8	6.6	0	0.35	0.35	1.2	2.1	5.6
v_5	7.1	5.0	6.4	1.6	0	0.35	1.2	2.1	5.6
v_6	6.5	4.4	7.2	1.0	1.4	0	0.85	1.6	5.6
v_7	6.8	4.7	7.7	1.3	1.7	1.2	0	2.1	6.1
v_8	7.2	5.1	8.1	1.7	2.0	1.6	0.4	0	5.3
v_9	13	11	11	7.7	6.1	6.6	6.4	6.0	0

From Table 1 it can be seen that the distance between two locations is not symmetric, i.e. the distance from node v_i to v_j may not be equal with the distance from v_j to v_i.

3. RESULTS AND DISCUSSION

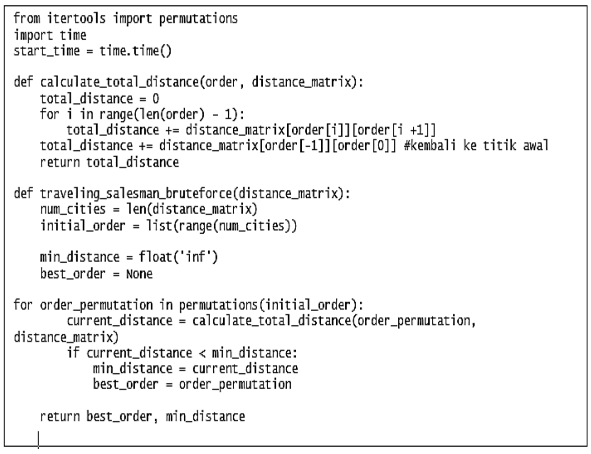
Implementation Using Brute Force

Solving TSP using the Brute-Force algorithm is completed by calculating all possible paths and finding the distance from all these possibilities. Because the TSP constitutes a tour, the number of possible tours is 8! = 40320 tours. Table 2 below shows some of the possibilities.

Table 2 Some Possible Tours

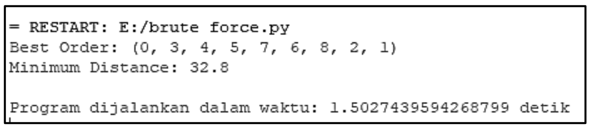
No	Tour	Total distance
1	1-2-3-4-5-6-7-8-9-1	37.25 km
2	1-2-3-4-5-6-7-9-8-1	36.15 km
3	1-2-3-4-5-6-9-7-8-1	37.30 km
4	1-2-3-4-5-9-6-7-8-1	38.00 km
5	1-2-3-4-9-5-6-7-8-1	37.50 km
6	1-2-3-9-4-5-6-7-8-1	37.65 km
7	1-2-9-3-4-5-6-7-8-1	40.95 km
8	1-9-2-3-4-5-6-7-8-1	46.25 km
9	1-2-3-4-5-6-8-7-9-1	37.10 km
10	1-2-3-4-5-8-6-7-9-1	39.30 km
11	1-2-3-4-8-5-6-7-9-1	39.70 km
12	1-2-3-8-4-5-6-7-9-1	39.75 km
13	1-2-8-3-4-5-6-7-9-1	44.35 km
14	1-8-2-3-4-5-6-7-9-1	46.15 km
15	1-2-3-4-5-7-6-8-9-1	37.95 km
...	...	...
40,320	9-8-7-6-5-4-3-2-1-9	36.30 km

Figure 1 shows the screenshot of the source code that is written in Phyton programming language, while Figure 2 shows the result. In the source code, the nodes are labelled as 0, 1, 2, 3, ..., 8. Thus, in the source code Mall Ciplaz (v_1) is assigned as 0, and Lampung City Mall (v_9) is assigned or labelled as 8.



```
from itertools import permutations
import time
start_time = time.time()

def calculate_total_distance(order, distance_matrix):
    total_distance = 0
    for i in range(len(order) - 1):
        total_distance += distance_matrix[order[i]][order[i +1]]
    total_distance += distance_matrix[order[-1]][order[0]] #kembali ke titik awal
    return total_distance

def traveling_salesman_bruteforce(distance_matrix):
    num_cities = len(distance_matrix)
    initial_order = list(range(num_cities))

    min_distance = float('inf')
    best_order = None

for order_permutation in permutations(initial_order):
        current_distance = calculate_total_distance(order_permutation,
distance_matrix)
        if current_distance < min_distance:
            min_distance = current_distance
            best_order = order_permutation

    return best_order, min_distance
```

Figure 1 Part of the Source Code for Brute Force



```
= RESTART: E:/brute force.py
Best Order: (0, 3, 4, 5, 7, 6, 8, 2, 1)
Minimum Distance: 32.8

Program dijalankan dalam waktu: 1.5027439594268799 detik
```

Figure 2 The Output for Brute Force Algorithm

Implementation Using Cheapest Insertion Heuristic

To solve the TSP using the data manually, first make a subtour from v_1 - v_2 - v_1 or v_2 -v_1 - v_2. Then, make a table that shows the smallest possible node to be added in the tour to make a new and bigger subtour. Table 3 shows all possible values.

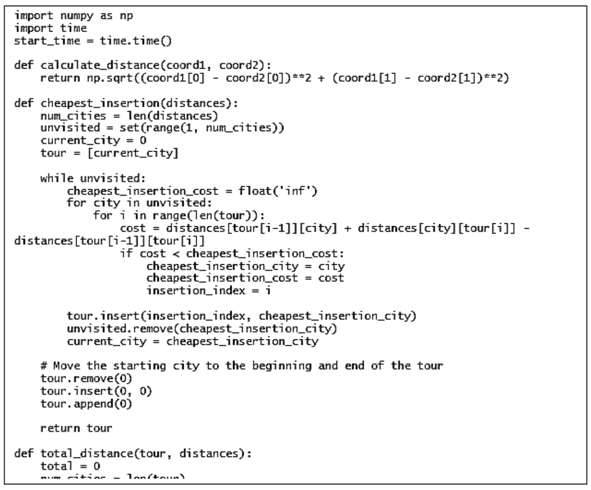
Table 2 Example to Show How to Form the New Subtour

Candidate Edge to be Chosen	Added Value	Selected Edge	Deleted Edge	New Subtour
e_{13}	$c_{13} + c_{32} - c_{12} = 8.4$			
e_{14}	$c_{14} + c_{42} - c_{12} = 7.8$			
e_{15}	$c_{15} + c_{52} - c_{12} = 8.7$			
e_{16}	$c_{16} + c_{62} - c_{12} = 7.8$			
e_{17}	$c_{17} + c_{72} - c_{12} = 8.9$			
e_{18}	$c_{18} + c_{82} - c_{12} = 10.1$			
e_{19}	$c_{19} + c_{92} - c_{12} = 19.1$	e_{23}	e_{21}	$v_2 - v_3 - v_1 - v_2$
e_{23}	$c_{23} + c_{31} - c_{21} = 5.5$			
e_{24}	$c_{24} + c_{41} - c_{21} = 7.4$			
e_{25}	$c_{25} + c_{51} - c_{21} = 8$			
e_{26}	$c_{26} + c_{61} - c_{21} = 7.8$			
e_{27}	$c_{27} + c_{71} - c_{21} = 8.6$			
e_{28}	$c_{28} + c_{81} - c_{21} = 9.9$			
e_{29}	$c_{29} + c_{91} - c_{21} = 19.2$			

e_{ij} is the edge that connect node i and node j, and c_{ij} is the distance between node i and node j. Because the distance from node v_i to v_j may not be equal with the distance from v_j to v_i, we consider for the first step to make tour v_1 - v_2 - v_1 and v_2 -v_1 - v_2.

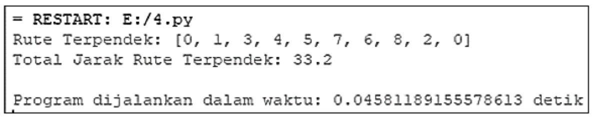
Table 3 shows all candidate edges to be selected to create a new subtour after subtour v_1 - v_2 - v_1 or v_2 -v_1 - v_2 created. From that table, the smallest added value is 5.5 by adding/inserting edge e_{23} and removing edge e_{21}, and get a new subtour $v_2 - v_3 - v_1 - v_2$. That process is repeated for inserting other vertices until a tour that consists all vertices is found.

Figure 3 shows parts of the source code for Cheapest Insertion Heuristic.



```
import numpy as np
import time
start_time = time.time()

def calculate_distance(coord1, coord2):
    return np.sqrt((coord1[0] - coord2[0])**2 + (coord1[1] - coord2[1])**2)

def cheapest_insertion(distances):
    num_cities = len(distances)
    unvisited = set(range(1, num_cities))
    current_city = 0
    tour = [current_city]

    while unvisited:
        cheapest_insertion_cost = float('inf')
        for city in unvisited:
            for i in range(len(tour)):
                cost = distances[tour[i-1]][city] + distances[city][tour[i]] -
distances[tour[i-1]][tour[i]]
                if cost < cheapest_insertion_cost:
                    cheapest_insertion_city = city
                    cheapest_insertion_cost = cost
                    insertion_index = i

        tour.insert(insertion_index, cheapest_insertion_city)
        unvisited.remove(cheapest_insertion_city)
        current_city = cheapest_insertion_city

    # Move the starting city to the beginning and end of the tour
    tour.remove(0)
    tour.insert(0, 0)
    tour.append(0)

    return tour

def total_distance(tour, distances):
    total = 0
```

Figure 3 Part of the Source Code for Cheapest Insertion Heuristic

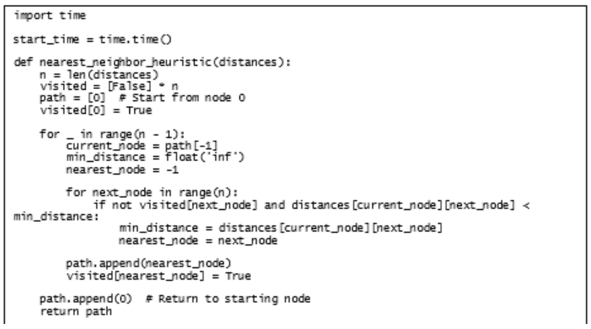


```
= RESTART: E:/4.py
Rute Terpendek: [0, 1, 3, 4, 5, 7, 6, 8, 2, 0]
Total Jarak Rute Terpendek: 33.2

Program dijalankan dalam waktu: 0.04581189155578613 detik
```

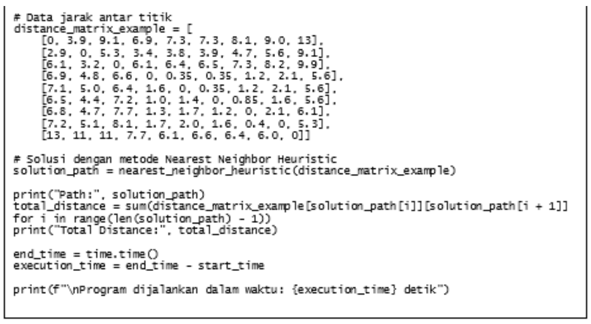
Figure 4 The Output for Cheapest Insertion Heuristic

Implementation Using Nearest Neighbor Heuristic

To use the Nearest Neighbor Heuristics, the first step is to determine the starting node, in this case Ciplaz, as node 1. After that, from the starting node, the nearest mall that has not been visited is selected as the next destination, in this case it is node 2. Next, the nearest node selection process is repeated in similar manner until all nodes are visited once.

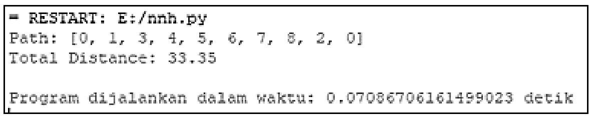


```
import time

start_time = time.time()

def nearest_neighbor_heuristic(distances):
    n = len(distances)
    visited = [False] * n
    path = [0]  # Start from node 0
    visited[0] = True

    for _ in range(n - 1):
        current_node = path[-1]
        min_distance = float('inf')
        nearest_node = -1

        for next_node in range(n):
            if not visited[next_node] and distances[current_node][next_node] <
min_distance:
                min_distance = distances[current_node][next_node]
                nearest_node = next_node

        path.append(nearest_node)
        visited[nearest_node] = True

    path.append(0)  # Return to starting node
    return path
```



```
# Data jarak antar titik
distance_matrix_example = [
    [0, 3.9, 9.1, 6.9, 7.3, 7.3, 8.1, 9.0, 13],
    [2.9, 0, 5.3, 3.4, 3.8, 3.9, 4.7, 5.6, 9.1],
    [6.1, 3.2, 0, 6.1, 6.4, 6.5, 7.3, 8.2, 9.9],
    [6.9, 4.8, 6.6, 0, 0.35, 0.35, 1.2, 2.1, 5.6],
    [7.1, 5.0, 6.4, 1.6, 0, 0.35, 1.2, 2.1, 5.6],
    [6.5, 4.4, 7.2, 1.0, 1.4, 0, 0.85, 1.6, 5.6],
    [6.8, 4.7, 7.7, 1.3, 1.7, 1.2, 0, 2.1, 6.1],
    [7.2, 5.1, 8.1, 1.7, 2.0, 1.6, 0.4, 0, 5.3],
    [13, 11, 11, 7.7, 6.1, 6.6, 6.4, 6.0, 0]]

# Solusi dengan metode Nearest Neighbor Heuristic
solution_path = nearest_neighbor_heuristic(distance_matrix_example)

print("Path:", solution_path)
total_distance = sum(distance_matrix_example[solution_path[i]][solution_path[i + 1]]
for i in range(len(solution_path) - 1))
print("Total Distance:", total_distance)

end_time = time.time()
execution_time = end_time - start_time

print(f"\nProgram dijalankan dalam waktu: {execution_time} detik")
```

Figure 5 The Source Code for Nearest Neighbor Heuristic



```
= RESTART: E:/nnh.py
Path: [0, 1, 3, 4, 5, 6, 7, 8, 2, 0]
Total Distance: 33.35

Program dijalankan dalam waktu: 0.07086706161499023 detik
```

Figure 6 The Output for Nearest Neighbor Heuristic

4. CONCLUSION

Based on the discussion above, we can conclude that the best solution for the problem of visiting nine malls in Bandar Lampung, starting at any mall, and then visiting the other eight malls exactly once and returning to the starting location, is gained by Brute Force algorithm with total distance 32.8 km and tour 1-4-5-6-8-7-9-3-2-1 or Ciplaz – Ramayana Tanjung Karang – Chandra Pasar Bawah – Simpur Center – Central Plaza – Mall Kartini – Lampung City Mall – Transmart Bandar Lampung – Mall Boemi Kedaton – Ciplas. The solution gained by using the *Cheapest-Insertion Heuristic* (CIH) is 33.2 km (which is the same solution gained manually and using Phyton programming language with tour 1-2-4-5-6-8-7-9-3-1 or Ciplaz – Mall Boemi Kedaton – Ramayana Tanjung Karang – Chandra Pasar Bawah – Simpur Center – Central Plaza – Mall Kartini – Lampung City Mall – Transmart Bandar Lampung – Ciplaz..

The solution gained by the *Nearest-Neighbor Heuristic* is 33.35 km with tour 1-2-4-5-6-7-8-9-3-1 or Ciplaz – Mall Boemi Kedaton – Ramayana Tanjung Karang – Chandra Pasar Bawah – Simpur Center – Mall Kartini – Central Plaza – Lampung City mall – Transmart Bandar Lampung – Ciplaz. The solutions yielded by Pthyton language programming and manually are the same solution). Therefore, in this case, visiting nine malls in Bandar Lampung exactly once and returning to the origin, the Brute Force algorithm provides the best solution with 33.8 km, followed by the Cheapest Insertion Heuristic with 33.2 km, and the Nearest Neighbor Heuristic with 33.35 km.

REFERENCES

[1] K. Saleh, Helmi, & B. Prihandono, B. Penentuan Rute Terpendek Dengan Menggunakan Algoritma Cheapest Insertion Heuristic (Studi Kasus: PT. Wicaksana Overseas International Tbk. Cabang Pontianak). *Buletin Ilmiah Math. Stat. Dan Terapannya (Bimaster)*, *04*(3), 295–304, 2015.

[2] K. Rosen. *Discrete Mathematics and Its Applications, 7th Ed.* McGraw-Hill, New York, 2012

[3] E. L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization* John Wiley & sons, 1985.

[4] M. Kurniawan, F. Farida, and S. Agustini,"Rute Terpendek Algoritma Particle Swarm Optimization dan Brute Force untuk Optimasi Travelling Salesman Problem, Jurnal Teknik Infoematika, Vol. 14 No. 2, 191-200, 2021.

[5] S. Violina, "Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP)", *Turkish Journal of Computer and Mathematics Education*, *12*(8), 1226–1229, 2021.

[6] A. Wilson, Y. Sibaroni, and I. Ummah, "*Analisis Penyelesaian Traveling Salesman Problem Dengan Metode Brute Force Menggunakan Graphic Processing Unit*", *2*(1), 1874-1883, 2015.

[7] Kusrini and J. E., Istiyanto, J.E., "Penyelesain Travelling Salesman Problem dengan Algoritma Cheapest Insertion Heuristic dan Basis Data", *Jurnal Teknik Informatika*: **8**(2)109-114, 2007.

[8] F., Fargiana, R. Respitawulan, Y. Fajar, D. Suhaedi, and E. Harahap. "Implementation of Cheapest Insertion Heuristic Algorithm in Determining Shortest Delivery Route", *International Journal of Global Operations Research*, *3*(2), 37–45, 2022. https://doi.org/10.47194/ijgor.v3i2.137

[9] M. Hilman, and Y. Sidik. "Penentuan Rute Distribusi Menggunakan Metode Cheapest Insertion Heuristic (Cih) Guna Meminimalkan Pengeluaran Biaya Pada Ukm Aren Creativity Di Kabupaten Ciamis", *Jurnal Industrial Galuh*, *4*(2), 51–61,2023. https://doi.org/10.25157/jig.v4i2.3017

[10] D. T. Wiyanti, D. T. "Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem", *Jurnal Transformatika",11*(1),1-6,2013. https://doi.org/10.26623/transformatika.v11i1.76

[11] E. Yulianto, and A. Setiawan, A. "Optimasi Rute Sales Coverage Menggunakan Algoritma Cheapest Insertion Heuristic Dan Layanan Google Maps Api", *INTERNAL (Information System Journal)*, *1*(1), 39–54, 2018. https://doi.org/10.32627/internal.v1i1.326

[12] S.A. Nene, and S. K. Nayar. "*A Simple Algorithm for Nearest Neighbor Search in High Dimensions*", (Vol. 4, Issue 1, 1995. Department of Computer Science, Columbia University. IEEE Transaction on Pattern Analysis and Machine Intelligence", Vol. 19, 1, 989-1003, 1997.

[13] Sutoyo, I, " Penerapan Algoritma Nearest Neighbour untuk Menyelesaikan Travelling Salesman Problem", *Paradigma - Jurnal Komputer Dan Informatika*, *XX*(1), 101–106, 2018