Implementation of Modified Cheapest Insertion Heuristic on Generating Medan City Tourism Route

Dian Rachmawati¹* and Wilyanto¹*

¹Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia

*Email: dian.rachmawati@usu.ac.id and wilyanto@students.usu.ac.id

Abstract. Arranging the sequence of places to visit is the common problem that tourists or foreigners encountered when visiting a new site, especially for foreigners that visit Medan City. This kind of case similar to Traveling Salesman Problem cases. Therefore we could use one of the modified TSP algorithm to solve it, which is Modified Cheapest Insertion Heuristics. The Modified Cheapest Insertion is a method to insert the cheapest weight starting from 2 points of place until all of the sites are being connected from the starting point until the endpoint. The strength of Modified Cheapest Insertion Heuristics is the calculation will remain stable even for a large number of inputs. In this research, all of the desired-to-visit places are being connected as a route.

1. Introduction

Nowadays there are so many foreigners, or tourists visits Medan City. By 2011 until 2015, there were 197,818 visitors (Badan Pusat Statistik 2018). Due to the high number of visitors in Medan City, many foreigners will deal with the common problem as scheduling and finding the most optimal route to start with. Looking at the problem, it is mostly similar to the Traveling Salesman Problem. Traveling Salesman Problem (TSP) is the most famous and well-studied problem in the combinatorial optimization area [1]. Traveling Salesman Problem (TSP) is the problem of finding a shortest closed tour that visits all the cities in a given set [2].

The Insertion Heuristic is similar to the Cheapest Insertion Heuristic (CIH), which known for being fast, producing suitable solutions, simple to implement, and easy to extend handling complicated constraints [3]. The Cheapest Insertion Heuristic firstly introduced for the Traveling salesman problem and then later extended to solve its extension [4]. The advantage of Cheapest Insertion Heuristic is the algorithm is still stable, even getting with many inputs rather than the other’s [5]. The Cheapest Insertion algorithm is useful in traveling salesman problems and known as heuristic methods. In our version, the initial partial tour is a minimum cost 2-edge cycle [6]. Its approach has two stages of completion: (1) the initialization stage, and (2) iteration stage. At the initialization stage, a route is formed, which starts and ends at the same point, namely depot available for any vehicle while the iteration stage is a stage for calculating profitability, checking feasibility, and inserting places that are not on the route [7].

2. Method

In this implementation, we need the user's input (e.g., places' name) and convert it to Longitude and Latitude coordinates, the distance between each location, and applying modified Cheapest Insertion Heuristics.
2.1 Steps for Generating each Places’ Longitude and Latitude Coordinates

2.1.1 Get All Destination as Input

The users need to input the starting point, the destination to visit, and where they want to go back as their last destination.

List of the inputs:

Starting point: Hotel Adimulia
Destinations: Tjong A Fie Mansion, Sun Plaza, Merdeka Walk, Istana Maimun
Last goal of the day: Hotel JW Marriot

| Arc Variable | Place Name          |
|--------------|---------------------|
| s (starting point) | Hotel Adimulia         |
| f (finishing point)   | Hotel JW Marriot     |
| d1 (Destination 1)       | Tjong A Fie Mansion |
| d2 (Destination 2)       | Sun Plaza            |
| d3 (Destination 3)       | Merdeka Walk         |
| d4 (Destination 4)       | Istana Maimun        |

| Table 1. Assign Place Name to Variables |

Here is the display of all location's pinpoint represents in Google Maps as shown below

![Google Maps display](image)

Figure 1. Displaying All Location's PinPoint

2.1.2 Get All Destinations’ Longitude and Latitude Coordinate

After we have the name of each place, we are going to get its longitude and latitude coordinate using [https://www.mapcoordinates.net/en](https://www.mapcoordinates.net/en) feature.
Example to get Sun Plaza’s longitude and latitude coordinate:

![Sun Plaza's Coordinates](image)

**Figure 2.** Getting Longitude and Latitude Coordinates

As we can see, the coordinate of Sun Plaza is 3.5834496 as its longitude and 98.6714264 as its latitude. The same method applied for the rest of the inputs to get their coordinates, as shown below.

| Arc Variable | Place Name          | Longitude  | Latitude       |
|--------------|---------------------|------------|----------------|
| s (starting point) | Hotel Adimulia     | 3.58508515 | 98.67245968    |
| f (finishing point)  | Hotel JW Marriott  | 3.59624489 | 98.67578069    |
| d1 (Destination 1)  | Tjong A Fie Mansion| 3.58559825 | 98.68056758    |
| d2 (Destination 2)  | Sun Plaza          | 3.5834496  | 98.6714264     |
| d3 (Destination 3)  | Merdeka Walk       | 3.5896736  | 98.67820424    |
| d4 (Destination 4)  | Istana Maimun      | 3.57525695 | 98.68390988    |

### 2.2 Steps for Generating Distance of Each Place

After we have all the place's longitude and latitude coordinates, we can find the distance from one's place against another place by using HTTP Request through Google Distance Matrix API. And an example of HTTP Request has been shown below using:

https://maps.googleapis.com/maps/api/distancematrix/json?units=metric&origins=3.58508515%2C98.67245968%2C3.59624489%2C98.67578069%2C3.58559825%2C98.68056758%2C3.5834496%2C98.6714264%2C3.5896736%2C98.67820424%2C3.5808515%2C98.67245968%2C3.59624489%2C98.67578069%2C3.58559825%2C98.68056758%2C3.5834496%2C98.6714264%2C3.5896736%2C98.67820424&destinations=3.58508515%2C98.67245968%2C3.59624489%2C98.67578069%2C3.58559825%2C98.68056758%2C3.5834496%2C98.6714264%2C3.5896736%2C98.67820424%2C3.5808515%2C98.67245968%2C3.59624489%2C98.67578069%2C3.58559825%2C98.68056758%2C3.5834496%2C98.6714264%2C3.5896736%2C98.67820424&key=AIzaSyBzT59uBINGxN5qGFK8ZSf6eStniy97W8E

Our HTTP Request at the above has query parameters, as shown below.
Figure 3. HTTP Request Query Parameters

Units: Distance Matrix results contain text within distance fields to indicate the distance of the calculated route. (units=metric will return distances in kilometers and meters).

Origins: The starting point for calculating travel distance and time. You can supply one or more locations separated by the pipe character (|) in the form of an address, latitude/longitude coordinates, or a place ID.

Destinations: One or more locations to use as the finishing point for calculating travel distance and time. The options for the destinations parameter are the same as for the origins parameter, described above.

Key: Our application's API key. This key identifies our application for purposes of quota management.

After we make an HTTP Request, now we will get feedback in JSON and XML responses, as shown below.

```

units:metric

origins:3.58505855 , 98.67240889 | 3.58624489 , 98.68578889 | 3.58555825 , 98.68865758 | 3.5834496 , 98.6714264 | 3.5896736 , 98.67820424 | 3.57525695 , 98.66399888

destinations:3.58505855 , 98.67240889 | 3.58624489 , 98.68578889 | 3.58555825 , 98.68865758 | 3.5834496 , 98.6714264 | 3.5896736 , 98.67820424 | 3.57525695 , 98.66399888

key:AIzaSyRzT50uh1NG6x8DqGkEdZ5f6eStnny9tWBE
```

Figure 4. HTTP Request Responses
The JSON and XML responses still go on for about six rows that contain six elements in each row due to our request using six origins and six destinations. As we can get the distance (in meters) from one to another's place from the responses, we store it to variables, and it could be represented to the table shown below.

|     | s     | f     | d1    | d2    | d3    | d4    |
|-----|-------|-------|-------|-------|-------|-------|
| S   | 0     | 1956  | 2028  | 1758  | 1922  | 3461  |
| F   | 2558  | 0     | 2840  | 2570  | 2388  | 4273  |
| d1  | 1730  | 1526  | 0     | 1742  | 535   | 3445  |
| d2  | 1695  | 2711  | 2784  | 0     | 2677  | 4014  |
| d3  | 2392  | 991   | 1794  | 2404  | 0     | 2579  |
| d4  | 3188  | 2796  | 1270  | 3200  | 1806  | 0     |

**Table 3. Distances within One Place to the Another’s**

### 2.3 Steps for Generating Route by Using Modified Cheapest Insertion Heuristics

This case similar to the Traveling Salesman Problem, therefore Cheapest Insertion Heuristics is a well-known heuristic to solve a case that similar to Traveling Salesman Problem [8].

Based on Wayne Winston's paper, here are the Cheapest Insertion Heuristic steps [9]:

1. The route starts at the starting point and is connected to the last (the starting point itself)
2. Generate a sub tour between those 2 points (place). The sub tour itself is the route from the starting point going to all destinations and ended at the finishing point. e.g : (s, d4) → (d4, d2) → (d2, d3) → (d3, d1) → (d1, f)
3. Change one of the arc (route/way) between 2 places with the combinations method, i.e., we could do insertion on arc (i,j) with arc (i,k) and arc (k,j), in term of k's place is not a member of the sub tour. With all of the combinations, we choose the cheapest one. The newest Subtour could be :
   \[ C_{i,k} + C_{k,j} - C_{i,j} \]
   whereas
   \[ C_{i,k} \] is the distance from the place i to place k
   \[ C_{k,j} \] is the distance from the place k to place j
   \[ C_{i,j} \] is the distance from the place i to place j

4. Repeat step number 3 until all the places are included in the sub tour.

But in this research, we need to make some modifications due to Cheapest Insertion Heuristics implement TSP. Because in our case, we can have different places between the starting point and finishing point, therefore we need to change some Cheapest Insertion Heuristics Algorithm rules about the finishing point. Here are the changes in the first step for modified Cheapest Insertion Heuristics steps :

1. The route starts at the beginning (starting) point and is connected to the last (finishing) point.
2. Generate a sub tour between those 2 points (place). The sub tour itself is the route from the starting point going to all destinations and ended at the finishing point. e.g : (s, d4) → (d4, d2) → (d2, d3) → (d3, d1) → (d1, f)
3. Change one of the arc (route/way) between 2 places with the combinations method, i.e., we could do insertion on arc (i,j) with arc (i,k) and arc (k,j), in term of k's place is not a member of the sub tour. With all of the combinations, we choose the cheapest one. The newest Subtour could be :
   \[ C_{i,k} + C_{k,j} + C_{i,j} \]
   whereas
   \[ C_{i,k} \] is the distance from the place i to place k
   \[ C_{k,j} \] is the distance from the place k to place j
Ci,j is the distance from the place i to place j

4. Repeat step number 3 until all the places are included into the sub tour.

Now, to generate the shortest path through 6 places, as shown in Table 3, do these steps:
1. Take the route from s place and connect it to the f place.
2. Create a new sub tour that consists of (s, f)
3. Create a new table that consists of the value of each insertion applied to all the combinations as shown in Table 4.

| Arc to be replaced | Arc to be inserted to Subtour | Additional distance (meters) |
|-------------------|-------------------------------|-----------------------------|
| (s, f)            | (s, d1) → (d1, f)             | $C_{s,d1} + C_{d1,f} - C_{s,f} = 1598$ |
| (s, f)            | (s, d2) → (d2, f)             | $C_{s,d2} + C_{d2,f} - C_{s,f} = 2513$ |
| (s, f)            | (s, d3) → (d3, f)             | $C_{s,d3} + C_{d3,f} - C_{s,f} = 957$ |
| (s, f)            | (s, d4) → (d4, f)             | $C_{s,d4} + C_{d4,f} - C_{s,f} = 4301$ |

From Table 4, we can get the cheapest weight of additional distance if we change arc (s, f) become (s, d3) and (d3, f). Therefore, the newest sub tour become:

(s, d3) → (d3, f).

4. Create a new table again to get the cheapest additional distance until all the places are included in the sub tour (places that are not inside the sub tour is d1, d2, d4)

Current sub tour : (s, d3) → (d3, f)

| Arc to be replaced | Arc to be inserted to Subtour | Additional distance (meters) |
|-------------------|-------------------------------|-----------------------------|
| (s, d3)           | (s, d1) → (d1, d3)            | $C_{s,d1} + C_{d1,d3} - C_{s,d3} = 641$ |
| (s, d3)           | (s, d2) → (d2, d3)            | $C_{s,d2} + C_{d2,d3} - C_{s,d3} = 2513$ |
| (s, d3)           | (s, d4) → (d4, d3)            | $C_{s,d4} + C_{d4,d3} - C_{s,d3} = 3345$ |
| (d3, f)           | (d3, d1) → (d1, f)            | $C_{d3,d1} + C_{d1,f} - C_{d3,f} = 2329$ |
| (d3, f)           | (d3, d2) → (d2, f)            | $C_{d3,d2} + C_{d2,f} - C_{d3,f} = 4122$ |
| (d3, f)           | (d3, d4) → (d4, f)            | $C_{d3,d4} + C_{d4,f} - C_{d3,f} = 4384$ |

The cheapest additional distance is : (s, d1) → (d1, d3) with total weight 641
Therefore, newest sub tour is : (s, d1) → (d1, d3) → (d3, f)
Table 6. The 3\textsuperscript{rd} Insertion to Subtour

| Arc to be replaced | Arc to be inserted to Subtour | Additional distance (meters) |
|--------------------|-------------------------------|-------------------------------|
| (s, d\textsubscript{1}) | (s, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{1}) | \(C_{s,d2} + C_{d2,d1} - C_{s,d1} = 2514\) |
| (s, d\textsubscript{1}) | (s, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{1}) | \(C_{s,d3} + C_{d3,d4} - C_{s,d1} = 2703\) |
| (d\textsubscript{1}, d\textsubscript{1}) | (d\textsubscript{1}, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{1}) | \(C_{d1,d2} + C_{d2,d3} - C_{d1,d3} = 3884\) |
| (d\textsubscript{1}, d\textsubscript{3}) | (d\textsubscript{1}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{3}) | \(C_{d1,d4} + C_{d4,d3} - C_{d1,d3} = 4716\) |
| (d\textsubscript{3}, f) | (d\textsubscript{3}, d\textsubscript{2}) → (d\textsubscript{2}, f) | \(C_{d3,d2} + C_{d2,f} - C_{d3,f} = 4124\) |
| (d\textsubscript{3}, f) | (d\textsubscript{3}, d\textsubscript{4}) → (d\textsubscript{4}, f) | \(C_{d3,d4} + C_{d4,f} - C_{d3,f} = 4384\) |

The cheapest additional distance is : (s, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{1}) with additional distance 2514 meters.

Therefore, newest subtour is : (s, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{1}) → (d\textsubscript{1}, d\textsubscript{3}) → (d\textsubscript{3}, f).

Due to d\textsubscript{4} that still are not inside the sub tour. We always keep the iteration for all combinations that refer to d\textsubscript{4}'s insertion of each arc.

Table 7. The 4\textsuperscript{th} Insertion to Subtour

| Arc to be replaced | Arc to be inserted to Subtour | Additional distance (meters) |
|--------------------|-------------------------------|-------------------------------|
| (s, d\textsubscript{2}) | (s, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{2}) | \(C_{s,d4} + C_{d4,d2} - C_{s,d2} = 4903\) |
| (d\textsubscript{2}, d\textsubscript{1}) | (d\textsubscript{2}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{1}) | \(C_{d2,d4} + C_{d4,d1} - C_{d2,d1} = 2500\) |
| (d\textsubscript{1}, d\textsubscript{3}) | (d\textsubscript{1}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{3}) | \(C_{d1,d4} + C_{d4,d3} - C_{d1,d3} = 4716\) |
| (d\textsubscript{3}, f) | (d\textsubscript{3}, d\textsubscript{4}) → (d\textsubscript{4}, f) | \(C_{d3,d4} + C_{d4,f} - C_{d3,f} = 4384\) |

As we can see, the cheapest additional distance is 2500 for (d\textsubscript{2}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{1}) insertion. Therefore the sub tour will be : (s, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{1}) → (d\textsubscript{1}, d\textsubscript{3}) → (d\textsubscript{3}, f). Hence there is no destination that is not inside the sub tour, we could terminate the iteration, and the final answer is the sub tour itself.

3. Results and Discussions

In the end, all the places are included in the sub tour. Hence, we could get the shortest path by using modified Cheapest Insertion Heuristic is (s, d\textsubscript{2}) → (d\textsubscript{2}, d\textsubscript{4}) → (d\textsubscript{4}, d\textsubscript{1}) → (d\textsubscript{1}, d\textsubscript{3}) → (d\textsubscript{3}, f) with total distance:

\[
\begin{align*}
(s,d2) & \quad 1758 \\
(d2,d4) & \quad 4014 \\
(d4,d1) & \quad 1270 \\
(d1,d3) & \quad 535 \\
(d3,f) & \quad 991
\end{align*}
\]

\[
\text{Total distance (meters)} \quad 8568
\]

The route can be displayed in google maps as shown below.
4. Conclusions
In conclusion, the total distance to visit Tjong A Fie Mansion, Sun Plaza, Merdeka Walk, and Istana Maimun in terms of the starting point at Hotel Adimulia and finishing point at Hotel JW Marriot is 8568 meters with the route has been shown at figure 4.

5. Acknowledgments
The authors gratefully acknowledge that the present research is supported by Fund Dissemination IPTEKS Research Results for Lectures / Researchers of Universitas Sumatera Utara.

References
[1] Lawler E L and Lenstra J K 1985 The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Chichester: Wiley. 11.201-209
[2] Dorigo Marco, et al. 1996 Ant Colonies for the Traveling Salesman Problem Switzerland: Elsevier. Page 74
[3] Mester D. and O Bräysy 2007 Active-Guided Evolution Strategies for Large-Scale Capacitated Vehicle Routing Problems Computers & Operations Res., p. 2964-2975
[4] Solomon M M 1987 Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints Operations Res. p. 254-265
[5] Aristi, G. 2014 Perbandingan Algoritma Greedy, Algoritma Cheapest Insertion Heuristics, dan Dynamic Programming dalam Penyelesaian Traveling Salesman Problem”. Tasikmalaya: Jurnal Paradigma XVI 2 p. 57
[6] Tanjung W, Nurhasanah N, Suri Q and Supriyanto A 2017 Cheapest Insertion Heuristics Algorithm to Optimize WIP Product Distributions in FBS Fashion Industry Conf. Ser.: Mater. Sci. Eng. 277 012069
[7] Satyananda D and Wahyuningish S 2014 Characteristic Studies of Solution the Multiple Trip Vehicle Routing Problem (MTVRP) and its Application in Optimization of Distribution Problem”. Malang: Proc. of Int. Seminar on Mathematics Education and Graph Theory.
[8] Johnson D S and Papadimitriou C H 1977 Performance Guarantees for Heuristics, Ch. 5 in The Traveling Salesman Problem”. Chichester: Wiley & Sons.
[9] Winston W Land Goldberg J B 2004 Operations Research Application and Algorithms 4th Edition”. United States of America: Duxbury.