Traveling Salesman Problem Application for Multi-Stop Delivery Route Optimization

Angela Livia Arumsari - 13521094[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13521094@std.stei.itb.ac.id*

***Abstract*—The demand for delivery services has been increasing, especially during the pandemic. When choosing a delivery service, affordability and delivery time are the most important factors for customers. To offer the best price and quickest delivery time, delivery companies need an effective system for multi-stop route planning. This paper propose the traveling Salesman Problem algorithm as the solution in multi-stop delivery route optimization. This algorithm can find the shortest route from the multiple stops resulting in the most optimal route.**

***Keywords*—Delivery Services, Multi-Stop Delivery, Route Optimization, Traveling Salesman Problem**

I. INTRODUCTION

In this fast-paced moving world, technological development pushes people to become more and more connected. With the increasing connectivity of people from different places, the demand for goods delivery keeps increasing. This delivery happens every second between countries, cities, or maybe just districts. It's no wonder that this delivery concept is used in many available services.

Demand for delivery services keeps increasing especially during the pandemic. Based on a quick survey from MarkPlus, Inc, 39% of respondents from around Indonesia reported that their use of package delivery services significantly increased during the pandemic [1]. From the survey, it could be concluded two things that the customers take into consideration the most when choosing delivery services. Those two things are affordability and delivery time.

Food delivery service is one of the most popular delivery services besides package delivery. People nowadays find it convenient to order food just through several clicks in applications. From the data collected by We Are Social, more than 50% of internet users around the world use food delivery services. Indonesia itself occupied the first place by having 77% of internet users using food delivery services.

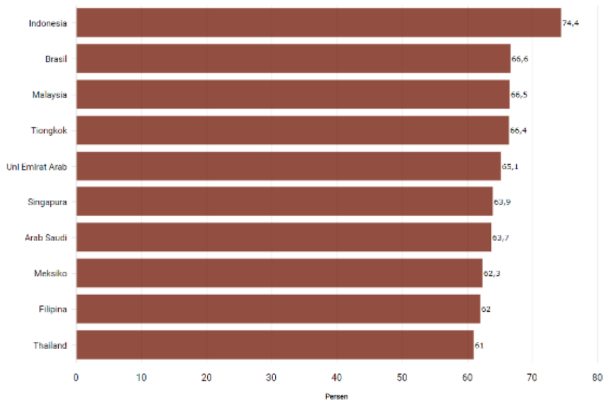
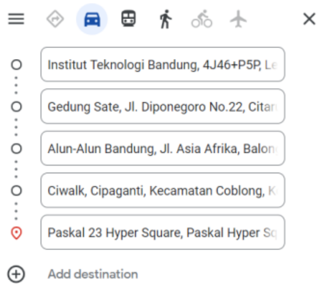


Fig. 1. Food Delivery User (Source: [2])

While the demand for delivery services is high, delivery companies need to have an effective system to be able to compete and offer the best price. One of the ways to have an effective system is to have optimal multi-stop route planning. With multi-stop delivery, companies could cut the unnecessary costs of travel to many destinations resulting in affordability for customers. With the most optimal route, companies could also cut the time needed to deliver resulting in quicker delivery.

A widely available free application for multi-stop delivery is Google Maps. In Google Maps, users can add up to 10 stops and the application will show the path from the starting point to each stop in order. While it's possible to search paths for multiple stops, Google Maps doesn't automatically show the quickest route. Users need to manually reorder stops to search for the most optimal route.



Institut Teknologi Bandung, 4J46+P5P, Le
Gedung Sate, Jl. Diponegoro No.22, Citar
Alun-Alun Bandung, Jl. Asia Afrika, Balon
Ciwalk, Cipaganti, Kecamatan Coblong, K
Paskal 23 Hyper Square, Paskal Hyper Sq
Add destination

Fig. 2. Google Maps with Multiple Stops (Source: Primary)

With the distance between stops as data, the quickest route of multiple stops delivery could be solved by the Traveling

Salesman Problem algorithm. In this paper, the author will solve multi-stop delivery route optimization using the Travelling Salesman Problem algorithm. The author will also compare the algorithm result with the result obtained from manually reordering stops from Google Maps.

II. Basic Theory

A. *Graph Definition*

A graph is a mathematical structure that represents the connections between discrete objects. It consists of vertices and edges. A vertex is a point in a graph that may or may not be connected to other vertices, and is typically represented by a dot or circle in a graph. An edge is a connection between two vertices and is represented by a line connecting two dots or circles.

Formally, a graph is represented as $G = (V, E)$, where V is a nonempty set of vertices, and E is a set of edges. Each edge has one or two vertices associated with it, called its endpoints. An edge connects its endpoints.

B. *Graph Types*

Based on the presence and absence of loops and multiple edges connecting the same vertices, the graph has two types:

1. Simple graphs
   Simple graphs are those in which each edge connects two different vertices and no two edges connect the same pair of vertices. In a simple graph, each edge is associated with an unordered pair of vertices and no other edge is associated with this same pair.

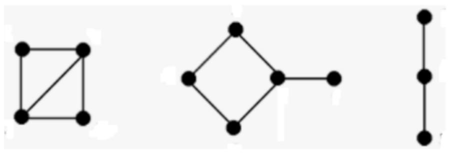


Fig. 3. Simple Graphs (Source: [4])

2. Unsimple graphs
   Unsimple graphs are those in which loops are present or multiple edges connect the same vertices. Unsimple graphs can be further divided into two subcategories:
   a. Multipgraphs
      Multipgraphs are graphs that have multiple edges connecting the same vertices. When there are m different edges associated with the same unordered pair of vertices $\{u, v\}$, then $\{u, v\}$ is an edge of multiplicity m.
   b. Pseudographs
      Pseudographs are graphs that may include loops and possibly multiple edges connecting to the same pair of vertices or a vertex to itself. Loops are edges that connect a vertex to itself.

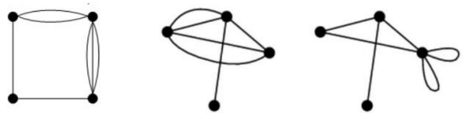


Fig. 4. Unsimple Graphs (Source: [4])

Based on the direction of the edges, graphs can be divided into two types:

1. Undirected graph
   In an undirected graph, each edge does not have any direction

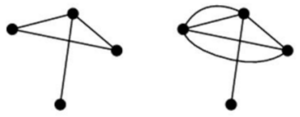


Fig. 5. Undirected Graphs (Source: [4])

2. Directed graph
   A directed graph (V, E) consists of a nonempty set of vertices V and a set of directed edges E. Each directed edge is associated with an ordered pair of vertices. Edge with the ordered pair $\{u, v\}$ is an edge that starts at u and ends at v.

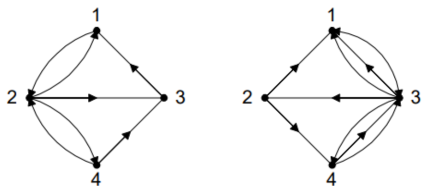


Fig. 6. Directed Graphs (Source: [4])

C. *Graph Terminologies*

In graph theory, there are several terms used when analyzing graphs. The terminologies that will be used are as follows:

1. Adjacent
   In an undirected graph, two vertices are considered adjacent if they are connected by an edge. In a graph with directed edges, the vertex at the beginning of the edge is considered adjacent to the vertex at the end of the edge. When (u, v) is an edge of the graph G, u is the initial vertex of an and v is the terminal or end vertex. If an edge forms a loop, the initial and terminal vertices are the same.
2. Incidence
   An edge (u, v) that connects u and v is called an incident with the vertices u and v.
3. Degree
   In an undirected graph, the degree of a vertex is defined as the number of edges that are incident with it. If a loop is present at a vertex, it is counted twice in the degree of that vertex. The degree of a vertex is represented by the notation $deg(v)$. A vertex with no incident edges is referred to as an isolated vertex. When the entire graph consists solely of isolated vertices, it is called a null or empty graph.
4. Path
   A path is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along the edges of the graph. As the path progresses, it passes through the vertices that are the endpoints of these edges. Two vertices are considered connected if there is a path between them. A graph is said to be connected if every pair of vertices is connected. In the case of a directed graph, strong connectivity refers to a directed

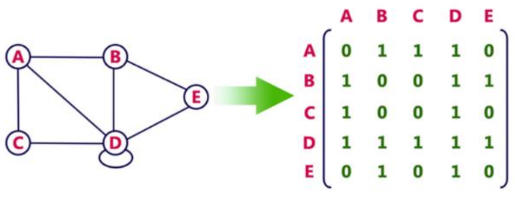
path from u to v and a directed path from v to u for every pair of vertices u, v. Weak connectivity, on the other hand, refers to a directed graph that produces a connected graph only if all of its directed edges are replaced with undirected ones.

5. Cycle
   A cycle is a sequence of vertices and edges that starts and ends at the same vertex. In other words, it is a path that does not have a specific starting or ending point but instead forms a closed loop. The length of a cycle is the number of edges it contains, and a graph can have multiple cycles of varying lengths.
6. Subgraph
   A subgraph is a subset of the vertices and all the edges of a larger graph. This subset forms a smaller graph that maintains the same properties and connections as the original graph.
7. Weighted Graph
   A weighted graph is a graph that assigns weights, or numerical values, to each of its edges. These weights can represent a variety of things, such as distance, price, cost, or queue. A weighted graph can be implemented in both directed and undirected graphs.

D. Graph Representation

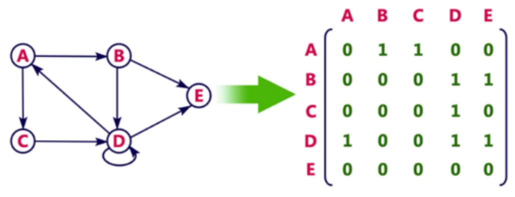
Graph representation is a method of storing graphs in the memory of a computer. This is achieved by using a set of vertices and identifying the neighbors of each vertex, which are the vertices directly connected to it by an edge. If the graph is weighted, the weight of each edge is also included in the representation. One of the best ways to represent graphs with a lot of edges is an adjacency matrix.

Suppose that $G = (V, E)$ is a simple graph where the amount of vertices is n. Suppose the vertices of G are listed in an arbitrary order as v_1, v_2, and so on. The adjacency matrix A of G, based on this ordering of the vertices, is an $n \times n$ matrix of zeros and ones, with a value of 1 in the a_{ij} entry if v_i and v_j are adjacent and 0 if they are not adjacent. In the case of a weighted graph, the weights of the edges can be stored instead of zeros and ones.



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 1 & 0 \\ B & 1 & 0 & 0 & 1 & 1 \\ C & 1 & 0 & 0 & 1 & 0 \\ D & 1 & 1 & 1 & 1 & 1 \\ E & 0 & 1 & 0 & 1 & 0 \end{array}$$

Fig. 7. Adjacency Matrix of Undirected Graph (Source: [5])



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 0 \\ B & 0 & 0 & 0 & 1 & 1 \\ C & 0 & 0 & 0 & 1 & 0 \\ D & 1 & 0 & 0 & 1 & 1 \\ E & 0 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 8. Adjacency Matrix of Directed Graph (Source: [5])

E. Hamiltonian Trail and Circuit

A Hamiltonian path is a path that traverses each vertice in a graph at least once. A Hamiltonian circuit, on the other hand, is a Hamiltonian path that ends at the starting point. This means that in a Hamiltonian circuit, all vertices are visited except for the starting point, which is visited twice. Graphs that possess only Hamiltonian paths are referred to as semi-Hamiltonian graphs, while those that contain Hamiltonian circuits are known as Hamiltonian graphs.

Sufficient conditions for a simple undirected graph to be a Hamiltonian graph is for n vertices with $n \geq 3$ in the graph to have at least $\frac{n}{2}$ degrees or more. However, it is possible for a graph to not satisfy this condition yet have a Hamiltonian circuit.

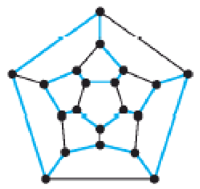


Fig. 9. Hamiltonian Circuit (Source: [3])

F. Traveling Salesman Problem

The traveling salesperson problem (TSP) is a well-known problem in computer science and operations research. Given a set of cities and the distances between each pair of cities, the goal of the TSP is to find the shortest possible tour that visits each city exactly once and returns to the starting city. This problem reduces to finding a Hamiltonian circuit in a complete graph such that the total weight of all the edges is as small as possible. The TSP has numerous applications in areas such as logistics, vehicle routing, and circuit design.

Many algorithms can be used to solve the traveling salesperson problem (TSP). One common approach is to use a heuristic algorithm, which is an algorithm that uses rules of thumb to find a good, but not necessarily the optimal solution to the problem. Some of the heuristic algorithms are explained below.

1. Nearest neighbor algorithm
   This algorithm begins by selecting an arbitrary starting city. It then repeatedly selects the nearest unvisited city and adds it to the tour. This process continues until all cities have been visited, at which point the algorithm returns to the starting city to complete the tour. In some cases, this algorithm can produce tours that are significantly longer than the shortest possible tour. This algorithm finds a solution in $O(N^2 log_2(N))$ iterations, where N is the number of cities to be visited. The nearest neighbor keeps the solution within 25% of the Held-Karp lower bound.
2. Genetic algorithm
   A genetic algorithm is a high-level algorithm that can be applied to a wide range of optimization problems. A genetic algorithm works by representing potential solutions to the TSP as strings of genes, where each gene represents a city in the tour. The genetic algorithm then evaluates the fitness of each tour in the population, using a fitness function that measures the quality of the

tour by using total distance. Based on the fitness of each tour, the algorithm selects the fittest tours to be included in the next generation of the population. This process of selection, crossover, mutation, and evaluation is repeated for several iterations, to find a high-quality solution to the TSP. This algorithm can find good solutions to the problem by exploring the space of possible tours. It can also handle large numbers of cities, but, it can be computationally expensive and may not always find the optimal solution to the problem. Applying a genetic algorithm to TSP requires certain limitations, such as, in every route, each city should not be repeated and only valid routes are considered in the algorithm.

3. Greedy heuristic algorithm
   The greedy heuristic algorithm begins by sorting all the edges and then selects the edge with the minimum cost. It continuously selects the best next choices given the condition that no loops are formed. The computational complexity of the greedy algorithm is $O(N^2 log_2(N))$ and there is no guarantee that a global optimum solution is found. On the other hand, the greedy algorithm terminates in a reasonable number of steps and keeps the solution within 15-20 % of the Held-Karplower bound.

III. Discussion

A. Limitations

There are many factors when applying the Travelling Salesman Problem to a multi-point delivery problem. However, in this paper, the author set some limitations to make the calculation simpler. The limitations are listed below.

1. Assuming that the shortest route is the quickest path for delivery.
2. The vehicle is assumed to take a car route and have enough fuel to complete the route until its last stop.
3. Suppose that starting point needs to be chosen, the greedy heuristic algorithm will be used in solving the problem. Even though the algorithm did not guarantee the most optimal solution, this algorithm is less consuming and slightly better than other heuristic algorithms.

B. Location Data

To calculate the most optimal route, the location of the stops needs to be represented as graphs. As samples, this paper uses eight locations in Bandung that act as the vertices for the graph representation. Those locations are listed in Table I below.

TABLE I
LOCATION DATA

Vertex	Location
1	Institut Teknologi Bandung
2	Unpad Dipatiukur
3	Gedung Sate
4	Ciwalk
5	Alun-alun Bandung
6	Universitas Katolik Parahyangan
7	Stasiun Bandung
8	Paskal 23 Hyper Square

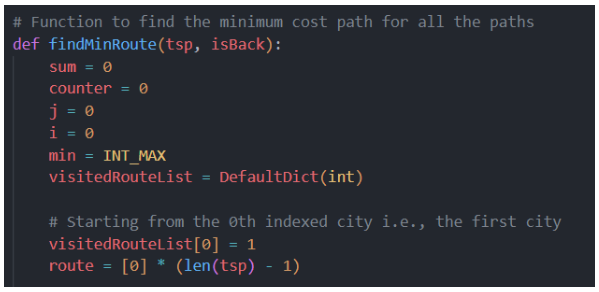
Each of the locations is connected to each other with different distances. This distance (in kilometers) will be the weight of the edge connecting two vertices. Distance between each location is collected from Google Maps. Because of the massive amount of edges, the graph will be represented by an adjacency matrix as seen in Figure 10 below.

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 0.85 & 1.7 & 3.8 & 4.7 & 3.3 & 3.5 & 4.8 \\ 2 & 1.2 & 0 & 1.2 & 3.8 & 5.3 & 3.4 & 4.0 & 5.3 \\ 3 & 1.7 & 1.7 & 0 & 4.9 & 3.9 & 4.4 & 3.1 & 4.9 \\ 4 & 1.9 & 2.4 & 2.6 & 0 & 4.8 & 4.8 & 2.9 & 4.2 \\ 5 & 4.7 & 4.7 & 3.5 & 5.4 & 0 & 7.4 & 1.4 & 2.0 \\ 6 & 3.5 & 3.4 & 4.8 & 2.4 & 7.0 & 0 & 5.2 & 6.5 \\ 7 & 3.8 & 3.8 & 2.9 & 5.5 & 1.9 & 6.5 & 0 & 0.95 \\ 8 & 3.8 & 4.9 & 4.1 & 4.0 & 2.6 & 6.2 & 0.75 & 0 \end{array}$$

Fig. 10. Adjacency Matrix of Location Graph (Source: Primary)

C. Traveling Salesman Problem Algorithm

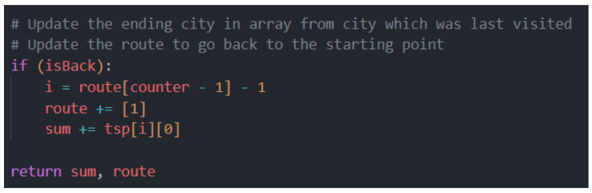
As mentioned in the limitations, the algorithm used to solve multi-stop delivery route optimization using TSP which is adapted from the greedy heuristic algorithm. The TSP algorithm accepts two parameters, tsp as the adjacency matrix of location graphs and isBack as a boolean. This boolean is needed to have an option on whether the route needs to go back to its starting point. This algorithm starts with initializing all variables needed as seen in Figure 11 below.



```
# Function to find the minimum cost path for all the paths
def findMinRoute(tsp, isBack):
    sum = 0
    counter = 0
    j = 0
    i = 0
    min = INT_MAX
    visitedRouteList = DefaultDict(int)

    # Starting from the 0th indexed city i.e., the first city
    visitedRouteList[0] = 1
    route = [0] * (len(tsp) - 1)
```

Fig. 11. TSP Algorithm Initialization (Source: Primary)

After initialization, the algorithm will start traversing the adjacency matrix of the location graphs. Within every traverse, the algorithm will check the distance between the current city and the unvisited city. The algorithm will choose the lowest distance from the current city and update the current city while adding the shortest distance city to the visited location. At the end of the traverse, if the isBack parameter is True, the algorithm will also add the distance from the last city to the starting city.



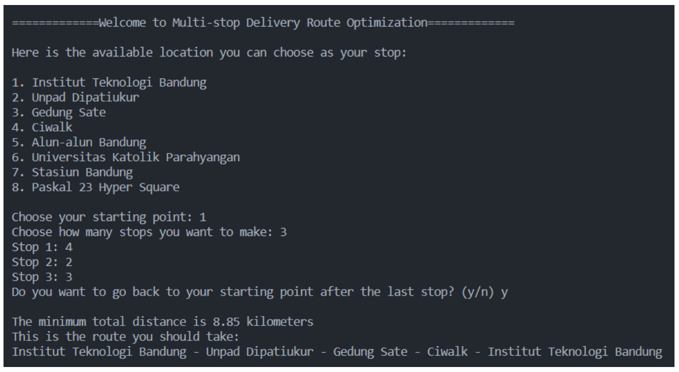
```
# Update the ending city in array from city which was last visited
# Update the route to go back to the starting point
if (isBack):
    i = route[counter - 1] - 1
    route += [1]
    sum += tsp[i][0]

return sum, route
```

Fig. 12. TSP Algorithm Back Option (Source: Primary)

This algorithm will return two values, the total of the most optimal distance and the most optimal route that should be taken. However, there is no guarantee that the algorithm will find the shortest distance because it only traverses the distance from the current city and chooses the shortest path every traverse.

D. Most Optimal Route Analysis

For the first case, vertex 1 will be chosen as the starting point. Suppose the delivery route needs to make 3 stops to vertex 4, 2, and 3 and it needs to go back to the starting point. In this case, the program will pass an adjacency matrix of vertex 1, 2, 3, and 4 as the location data for the TSP algorithm. It could be seen from Figure 13 below, the shortest distance is 8.85 kilometers while the shortest route is 1 - 2 - 3 - 4 - 1 or Institut Teknologi Bandung - Unpad Dipatiukur - Gedung Sate - Ciwalk - Institut Teknologi Bandung.



```
============Welcome to Multi-stop Delivery Route Optimization============

Here is the available location you can choose as your stop:

1. Institut Teknologi Bandung
2. Unpad Dipatiukur
3. Gedung Sate
4. Ciwalk
5. Alun-alun Bandung
6. Universitas Katolik Parahyangan
7. Stasiun Bandung
8. Paskal 23 Hyper Square

Choose your starting point: 1
Choose how many stops you want to make: 3
Stop 1: 4
Stop 2: 2
Stop 3: 3
Do you want to go back to your starting point after the last stop? (y/n) y

The minimum total distance is 8.85 kilometers
This is the route you should take:
Institut Teknologi Bandung - Unpad Dipatiukur - Gedung Sate - Ciwalk - Institut Teknologi Bandung
```

Fig. 13. First Case of Multi-Stop Delivery (Source: Primary)

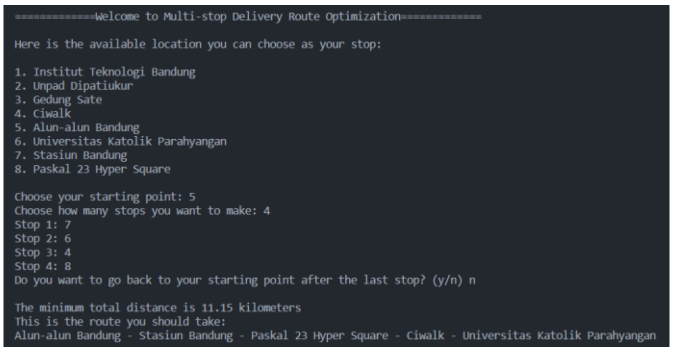
If Google Maps are used to find the most optimal route, multiple stops need to be added. Institut Teknologi Bandung is chosen as the starting point, while Unpad Dipatiukur, Gedung Sate, and Ciwalk. Institut Teknologi Bandung is added as the last stop because the delivery route needs to go back to its starting point. Google Maps will show the route based on the order of stops added. With a fixed starting point, there will be $3! = 6$ possible routes. By reordering stops manually, the data in the Table II below will be obtained.

TABLE II
GOOGLE MAPS DATA FOR THE FIRST CASE

Route	Total Distance (kilometers)	Time Estimation (minutes)
1 - 2 - 3 - 4 - 1	8.8	17
1 - 2 - 4 - 3 - 1	9.1	19
1 - 3 - 2 - 4 - 1	9.2	20
1 - 3 - 4 - 2 - 1	10	22
1 - 4 - 2 - 3 - 1	9.2	20
1 - 4 - 3 - 2 - 1	9.3	19

As can be seen from Table II, the quickest time to go through every stop is 17 minutes with the route "1 - 2 - 3 - 4 - 1". This matches the result obtained from the TSP algorithm. An interesting thing to note is that different distances can equal the same time needed. In the route "1 - 2 - 4 - 3 - 1" the total distance is 9.1 kilometers, while in the route "1 - 4 - 3 - 2 - 1" the total distance is 9.3 kilometers. However, they both have a time estimation of 19 minutes. It could also be seen that route "1 - 4 - 3 - 2 - 1" results in quicker time estimation than route "1 - 4 - 2 - 3 - 1" even though the distance is longer. This could be caused by unpredictable traffic conditions.

For the second case, vertex 5 will be chosen as the starting point. Suppose the delivery route needs to make 4 stops to vertex 7, 6, 4, and 8. In this route, the delivery doesn't need to go back to the starting point. In this case, the program will pass an adjacency matrix of vertex 4, 5, 6, 7, and 8 as the location data for the TSP algorithm. It could be seen from Figure 14 below, the shortest distance is 11.15 kilometers while the shortest route is "5 - 7 - 8 - 4 - 6" or "Alun-Alun Bandung - Stasiun Bandung - Paskal 23 Hyper Square - Ciwalk - Universitas Katolik Parahyangan".



```
============Welcome to Multi-stop Delivery Route Optimization============

Here is the available location you can choose as your stop:

1. Institut Teknologi Bandung
2. Unpad Dipatiukur
3. Gedung Sate
4. Ciwalk
5. Alun-alun Bandung
6. Universitas Katolik Parahyangan
7. Stasiun Bandung
8. Paskal 23 Hyper Square

Choose your starting point: 5
Choose how many stops you want to make: 4
Stop 1: 7
Stop 2: 6
Stop 3: 4
Stop 4: 8
Do you want to go back to your starting point after the last stop? (y/n) n

The minimum total distance is 11.15 kilometers
This is the route you should take:
Alun-alun Bandung - Stasiun Bandung - Paskal 23 Hyper Square - Ciwalk - Universitas Katolik Parahyangan
```

Fig. 14. Second Case of Multi-Stop Delivery (Source: Primary)

Using similar steps as before, multiple stops are added to Google Maps to obtain data from all the possible routes. With a fixed starting point and four stops, there will be $4! = 24$ possible routes. By manually reordering the stop order, the data present in Table III are obtained.

TABLE III
GOOGLE MAPS DATA FOR THE SECOND CASE

Route	Total Distance (kilometers)	Time Estimation (minutes)
5 - 4 - 6 - 7 - 8	17	39
5 - 4 - 6 - 8 - 7	17	41
5 - 4 - 7 - 6 - 8	21	49
5 - 4 - 7 - 8 - 6	17	39
5 - 4 - 8 - 6 - 7	21	41
5 - 4 - 8 - 7 - 6	17	41
5 - 6 - 4 - 7 - 8	14	32
5 - 6 - 4 - 8 - 7	15	33
5 - 6 - 7 - 4 - 8	21	47
5 - 6 - 7 - 8 - 4	18	41
5 - 6 - 8 - 4 - 7	21	46
5 - 6 - 8 - 7 - 4	20	46
5 - 7 - 4 - 6 - 8	18	42
5 - 7 - 4 - 8 - 6	17	41
5 - 7 - 6 - 4 - 8	14	33
5 - 7 - 6 - 8 - 4	18	43
5 - 7 - 8 - 4 - 6	11	30

5 - 7 - 8 - 6 - 4	11	28
5 - 8 - 4 - 6 - 7	16	37
5 - 8 - 4 - 7 - 6	15	37
5 - 8 - 6 - 4 - 7	14	31
5 - 8 - 6 - 7 - 4	17	41
5 - 8 - 7 - 4 - 6	13	32
5 - 8 - 7 - 6 - 4	12	28

As can be seen from Table III, the quickest time to go through every stop is 28 minutes with the route “5 - 7 - 8 - 6 - 4” and “5 - 8 - 7 - 6 - 4”. This doesn’t match the result from the TSP algorithm. The result from the algorithm is the route “5 - 7 - 8 - 4 - 6” with a time estimation of 30 minutes which is the second quickest route. The route “5 - 8 - 7 - 6 - 4” actually has a longer distance yet the time estimation is quicker. While the route “5 - 7 - 8 - 6 - 4” actually has a similar distance from the result yet the time estimation is also quicker. Unfortunately, this could be easily caused by the unpredictable traffic conditions on each route.

The result obtained from the algorithm is not always the most optimal. However, from Table II and Table III, it could be seen that the addition of one stop-point could add up massive possible route order. It would be ineffective for a delivery company to spend time and energy searching for the most optimal route. In contrast, the application of the Traveling Salesman Problem to multi-stop delivery route optimization will save time and resources. Even though this algorithm doesn't ensure the most optimal route, the result will still have an optimal route with a small margin of error.

IV. Conclusion

From this paper, it could be concluded that the Traveling Salesman Problem algorithm could be applied in multi-stop delivery route optimization. This algorithm can find the shortest route from the multiple stops a delivery should make. With the optimization of multi-stop delivery routes, delivery companies could make better offers for their customers. The optimization enables delivery companies to take the quickest route resulting in lower costs and quicker delivery.

This research could be developed by using a real-time location data API. Because of the limited time and resources, the Traveling Salesman Problem algorithm only used a small amount of location data with some limitations. However, using real-time data and time estimation as the weight of the edge, searching for the most optimal route will be more accurate.

V. Appendix

The complete Traveling Salesman Problem algorithm can be found below.
https://github.com/liviaarumsari/TSP-Multi-Point-Delivery.git

VI. Acknowledgment

This paper would not have been possible without the plenty of resources on Discrete Mathematics provided by all the lecturers of IF2120. The author would also like to especially thank Dr. Nur Ulfa Maulidevi, S.T., M.Sc. as the lecturer of class 01, for assigning this paper, as it is a way to learn the applications of the materials taught in class.

References

[1] A. N. Fitri, “Survei: Di Masa Pandemi, 85,2% Masyarakat Gunakan Jasa Kurir Untuk Pengiriman barang,” kontan.co.id, 20-Oct-2020. [Online]. Available: https://industri.kontan.co.id/news/survei-di-masa-pandemi-852-masyarakat-gunakan-jasa-kurir-untuk-pengiriman-barang. [Accessed Dec. 3, 2022].

[2] A. Lidwina, “Penggunaan aplikasi Pesan-Antar makanan Indonesia tertinggi di Dunia: Databoks,” *Pusat Data Ekonomi dan Bisnis Indonesia*, 18-Feb-2021. [Online]. Available: https://databoks.katadata.co.id/datapublish/2021/02/18/penggunaan-aplikasi-pesan-antar-makanan-indonesia-tertinggi-di-dunia. [Accessed Dec. 3, 2022].

[3] K. H. Rosen, Discrete Mathematics and Its Applications Seventh Edition. New York, America: McGraw-Hill, 2017.

[4] R. Munir, “Graf Bagian 1,” *IF2120 Matematika Diskrit*. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf. [Accessed Dec. 9, 2022].

[5] “Graph theory graph representations - javatpoint,” *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/graph-theory-graph-representations. [Accessed: Dec. 9, 2022].

[6] H. Abdulkarim and I. F. Alshammari, “Comparison of Algorithms for Solving Traveling Salesman Problem,” Current Issues in International Journal of Engineering and Advanced Technology, vol. 4, no. 6, August, 2015. [Online serial]. Available: https://www.researchgate.net/publication/280597707_Comparison_of_Algorithms_for_Solving_Traveling_Salesman_Problem. [Accessed Dec. 9, 2022].

[7] “Travelling salesman problem: Greedy Approach,” *GeeksforGeeks*, 05-Jan-2022. [Online]. Available: https://www.geeksforgeeks.org/travelling-salesman-problem-greedy-approach/. [Accessed: Dec. 9, 2022].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2022

[illegible]

Angela Livia Arumsari 13521094