Implementation of Hamiltonian Circuits on Efficient Waste Collection Route Planning on ITB Ganesha

Bevinda Vivian 13523120[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523120@std.stei.itb.ac.id

***Abstract*— Efficient waste management is a pressing challenge in crowded areas, including educational institutions like Bandung Institute of Technology (ITB). This study explores the application of Hamiltonian Circuits to optimize waste collection routes within ITB Ganesha. By addressing the Traveling Salesperson Problem (TSP), the proposed system generates the shortest route connecting all designated waste points while returning to the starting location. The implementation utilizes OpenRouteService to dynamically visualize optimized routes and pinpoint locations, ensuring all paths remain confined within ITB Ganesha campus boundaries.**

The system automatically assigns sequential numbers to waste collection points based on the optimal route and provides detailed information on total travel distance and estimated time, aiding drivers in executing waste collection tasks efficiently. This implementation not only minimizes travel distance and fuel consumption but also enhances sustainability and operational effectiveness. The results demonstrate a significant improvement in waste collection processes, serving as a model for broader applications in urban waste management and logistics optimization. Future enhancements could incorporate real-time traffic updates and machine learning for adaptive route planning.

Keywords—Hamiltonian Circuits, Traveling Salesperson Problem, waste management, route optimization, ITB Ganesha.

I. INTRODUCTION

Human habitation is formed, built, and inhabited by humans themselves. If not humans who maintain it, then who else? Today, I believe that most people know that environmental issues are a global issue. Becoming a global issue, it also impacts Indonesia's environment. One of the biggest ecological problems Indonesia faces is waste. Indonesia is facing difficulties that emerge from the people in Indonesia itself. Lastly, the newest issue is corruption in lean mining companies, this case causes ecological damage and the nation to lose around 300 trillion IDR. Sadly, this issue came from my hometown, Bangka Belitung. I still remember the last time I went back to my hometown and saw the beautiful scenery. But now my aunt has told me that it is very hot there and also not beautiful at all. What happened to this nation is an example of giving full attention to the environmental issue that we are facing in Indonesia.

Despite the heartbreaking events in Indonesia. There are also seeds of hope that the younger generation is aware of the importance of dealing with environmental issues. The most well-known example among the younger generation in Indonesia is the Pandawara Group, a group of five young people who have become influencers for young people to be aware of waste. They have tried to clean up highly polluted rivers throughout Indonesia, not only rivers but also sometimes rivers and riverbanks. Their efforts have not only helped address environmental issues in Indonesia but also raised awareness of the importance of proper waste management, especially for the younger generation. And relation to this, it has become proof that when people want to move, of course, this can affect the people around them too.

The rapid changes that occur in cities make the main problem in big cities like Jakarta or Bandung is waste management. If you are walking along a sidewalk in Jakarta or Bandung, public trash bins are rarely found. This is confusing, whether the government does not think about waste when managing the existing city planning. Of course, this is the root of the problem that can cause scattered waste which can later increase the very high risk of causing disease and many other problems. Regarding waste management in the city, especially in Bandung, the waste that has been collected will be transported more than 40 km to the Sarimukti landfill. Of course, this is also an example that waste management in the city should still be more efficient because if not, it will use up significant resources. To overcome and increase efficiency in managing waste, of course, innovative solutions are needed, whether it is related to resources or transportation.

No need to go far, as a student, I realize that the impact I can give can start from my campus. At ITB Ganesha, waste collection is managed through a structured system that includes sorting, optimal transportation, and the use of the Integrated Waste Processing Installation (IPST) located in Sabuga. IPST significantly increases the efficiency of waste processing, reduces costs, and generates income through partnerships with waste banks. By 2024, ITB aims to cut its waste management costs in half and further increase its income from recycling initiatives. However, if we look at it again to cut

management costs and increase the efficiency of waste management, it can easily be initiated from the waste collection at ITB itself.

Aligning with my studies in ITB, my major is Computer Science (Teknik Informatika), and Discrete Mathematics is one of the subjects in my major, I see opportunities to enhance ITB's waste collection route planning using computational approaches. With an interest in graph method in Discrete Mathematics. One of the graph theories I believe can make a big impact to enhance efficiency in waste collection, especially In ITB is the concept of Hamiltonian. I believe that the Hamiltonian circuit is particularly relevant for optimizing the route planning of waste collection in ITB Ganesha. This approach hopefully can reduce fuel consumption and enhance efficiency by minimizing the distances that would take to do the waste collection.

II. BASIC THEORY

A. Definition of Graph

A graph is a pair of sets (V,E) where V is a non-empty set of vertices (vertices/nodes) and E is a set of edges (edges/arcs) that connect a pair of vertices. *V* is the set of vertices (or nodes), representing objects. Represent as V = $\{v_1, v_2, v_3, \ldots, v_n\}$. *E* is the set of edges (or links), representing relationships or connections between vertices. Represent as E = $\{e_1, e_2, e_3, \ldots, e_n\}$. G represent as G = (V, E).

B. Types of Graph

Graphs can be categorized according to the existence or lack of loops or multiple edges, the count of vertices, and the direction's orientation. According to the existence or non-existence of loops or multiple edges, graphs can be categorized into:

1. Simple graph is a graph that lacks loops or multiple edges.
2. Unsimple graph is a graph that contains loops or multiple edges. A graph featuring several edges is referred to as a multigraph. A pseudograph is defined as a graph containing a ring edge.

According to their directional orientation, graphs can be categorized into:

1. Directed graphs, also known as graphs that possess a directional orientation.
2. Undirected graphs, meaning graphs that lack a directional orientation.

Graphs can be categorized according to the quantity of vertices into:

1. Finite graphs, specifically graphs that contain a limited number of vertices.
2. Infinite graphs, specifically graphs containing an infinite quantity of vertices.

C. Graph Terminology

1. Empty Graph

An empty graph is a graph that has no edges (its edge set is empty).

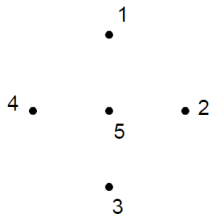

Fig 1. Empty Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025.

2. Adjacent

Two vertices in a graph are said to be adjacent if they are directly connected by an edge.

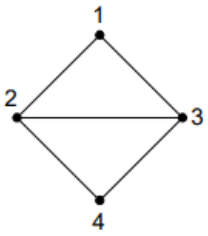

Fig 2. Adjacent Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025.

As shown in Fig. 2, vertex 1 is adjacent to vertex 2 and 3, but vertex 1 is not adjacent to vertex 4.

3. Incident

Edge E = (U,V) is said to be adjacent to vertices u and v. As shown in Fig. 2, E (2, 3) is adjacent to vertex 2 and vertex 3, E (2, 4) is adjacent to vertex 2 and vertex 4, but E (1, 2) is not adjacent to vertex 4.

4. Isolated Vertex

An isolated vertex is a vertex that does not have an edge adjacent to the vertex.

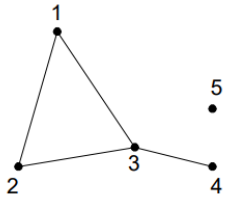

Fig 3. Isolated Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025.

As shown in Fig. 3, vertex 5 is an isolated vertex.

5. Degree

The degree of a vertex in an undirected graph is the number of edges adjacent to the vertex, represent as

d(v). The degree of a vertex in a directed graph is divided into two, namely first as in-degree $d_{in}(v)$, which is the number of arcs entering the vertex. The second is out-degree $d_{out}(v)$,, which is the number of arcs leaving the vertex.

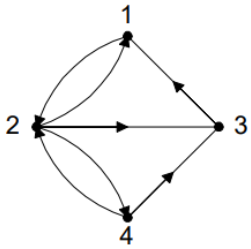

Fig 4. Directed Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025.

As shown in Fig. 3, it shows that d(1) = d(4) = 2 and d(2) = d(3) = 3. And shown in Fig. 4 $d_{in}(1) = 2$, $d_{out}(1) = 1$, $d_{in}(2) = 2$, $d_{out}(2) = 3$, $d_{in}(3) = 2$, $d_{out}(3) = 1$, $d_{in}(4) = 1$, $d_{out}(3) = 2$.

6. Path

The path from the starting node vo to the destination node vn in graph G is an alternating sequence of nodes and edges v0, e_1, v_1, e_2, … , v_{n-1}, e_n, v_n. Where $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, …, $e_n = (v_{n-1}, v_n)$.

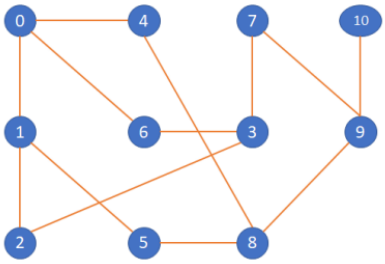

Fig 5. Unweighted Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025.

As shown in Fig. 5, it shows that path 0, 6, 3, 7, 9, 10 is a path from vertex 0 to 10 through edges (0, 6), (6,3), (3,7), (7, 9), (9, 10). The length of a path is the number of edges in the path. Path 0, 6, 3, 7, 9, 10 in G has a length of 5.

7. Cut-Set

A cut-set is a set of edges that, if removed from a graph G, will make the graph G disconnected.

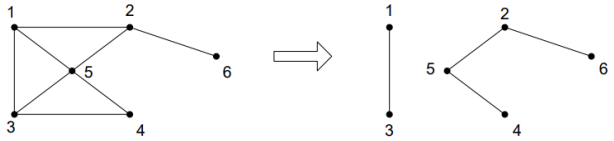

Fig 6. Unweighted Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

As shown in Fig. 6, it shows that if set of edges {(1,2), (1,5), (3,5), (3,4)} removed from the graph, it will make the graph disconnected. But, for set of edges {(1,2), (2,5), (4,5)} is not a cut-set, because the subset {(1,2), (2,5)} is a cut-set too.

8. Connected

A graph is said to be connected if for every pair of nodes u and v in the graph, there is a path from u and v. As shown in the Fig. 7, and Fig. 8 the difference berween connected and disconnected graph are illustrated.

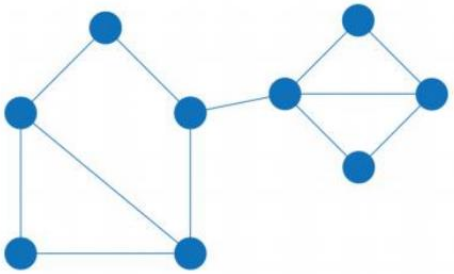

Fig 7. Connected Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

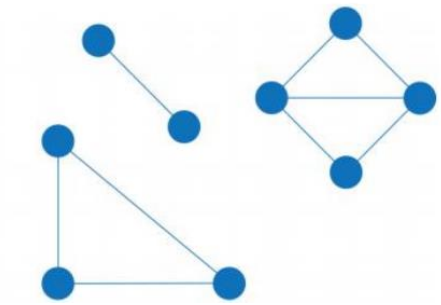

Fig 8. Disconnected Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

9. Subgraph

A graph A = (V_a, E_a) is called a subgraph of a graph G = (V, E) if $V_a \subseteq V$ and $E_a \subseteq E$. A subgraph illustrated in Fig. 9.

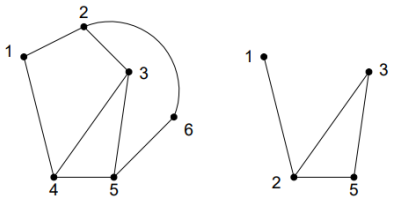

Fig 9. Graph and Its Subgraph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

10. Spanning Subgraph

A subgraph A = (V_a, E_a) is called a spanning subgraph if V_a = V. Spanning subgraph illustrated in Fig. 10.

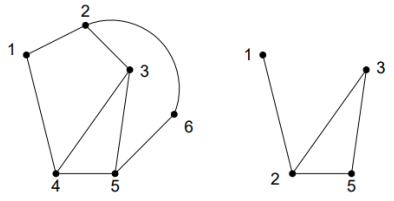

Fig 10. Graph and Its Subgraph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-

2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

11. Circuit

A circuit is a path that starts and ends at the same vertex. As shown in Fig. 5, it shows that path 0, 4, 8, 5, 1, 0 is a circuit. The length of a circuit is the amount of edges in the path. Path 0, 4, 8, 5, 1, 0 in G has a length of 5.

D. Hamiltonian Circuit and Travelling Salesperson Problem (TSP)

Hamiltonian Circuit is a circuit that passes through each vertex in the graph exactly once, except the origin (also end vertex) which is passed twice. A graph that has a Hamiltonian circuit is called a Hamiltonian graph. The Hamiltonian graph is illustrated in Fig. 11, has the path 1-2-3-4-1 that makes it a Hamilton graph.

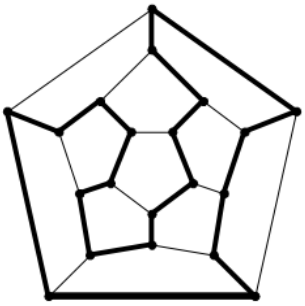

Fig 11. Hamiltonian Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, retrieved on 04/01/2025

The theorem that correlated to Hamiltonian circuit, the first one is a sufficient condition for a simple graph G with n (≥ 3) vertices to be a Hamiltonian graph is that the degree of each vertex is at least n/2 (that is, d(v) ≥ n/2 for every vertex v in G). The next theorem is every complete graph is a Hamiltonian graph. The next theorem is, in a complete graph G with n vertices (n ≥ 3), there are (n – 1)!/2 Hamilton circuits. The last theorem is, in a complete graph G with n vertices (n ≥ 3 and n is odd), there are (n – 1)/2 mutually exclusive Hamilton circuits (no intersecting sides). If n is even and n ≥ 4, then in G There are (n – 2)/2 Hamilton circuits which are mutually exclusive.

There exists a method to demonstrate that a graph is not Hamiltonian, a very straightforward approach allows us to examine each case for potential paths, proving that none succeed. For example, in Discrete Mathematics, we presented the proof that removing a vertex from a graph results in a non-Hamiltonian graph. A straightforward method to achieve this is to choose a vertex and examine the potential pairs of edges that the path could traverse through that vertex. For every option, we understand that certain edges will remain unused, allowing us to advance further in that direction. Typically, we aim to avoid brute-force case-by-case analyses as they can be challenging to ensure that all cases have been identified, and the proofs are not always insightful. It's significantly improved when we can discover a "reason" for why the graph lacks Hamiltonian properties. This approach to identifying causes frequently offers understanding of the graph's framework.

The Travelling Salesperson Problem (TSP) is a well-known optimization challenge in graph theory that is significantly linked to the Hamiltonian Circuit. In TSP, the goal is to determine the shortest route that visits every vertex (or city) precisely once and returns to the starting vertex. A Hamiltonian Circuit is concerned with the existence of the circuit, whereas TSP introduces the requirement of minimizing the overall weight (or cost) of the circuit. For instance, provided a set of cities along with the distances separating them, find the most efficient route that a salesperson should follow if they begin at a starting city, visit every city precisely one time, and return to the original city. This amounts to locating a Hamiltonian Circuit that has the least weight.

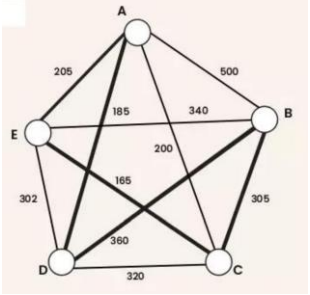

Fig 12. Hamiltonian Graph Illustration
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf, retrieved on 06/01/2025

As shown in Fig. 12, the shortest path that has minimum weight are A – D – B – C – E – A with the accumulation weight are 185 + 360 + 305 + 165 + 200 = 1220.

III. IMPLEMENTATION

A. Initialization

```python
import folium
import webbrowser
import openrouteservice
from random import uniform
import time

# konstanta
ITB_LOCATION = (-6.8915, 107.6107)
waste_points = [
    (f"Point {i+1}", uniform(-6.893, -6.889), uniform(107.609, 107.612))
    for i in range(50)
]
AVERAGE_SPEED = 15  # km/h

API_KEY = "5b3ce3597851110001cf62480807c7fdd6074c4a8a2135268b257005"
ors_client = openrouteservice.Client(key=API_KEY)
```

Fig 13. Initialization Code
Source: Author's Source Code

The implementation began by importing libraries essential for geospatial visualization, API communication, and computational tasks. The coordinates of ITB Ganesha

campus were defined as a central reference point, with 50 waste bin locations simulated within campus boundaries. Randomized coordinates were generated within a bounded area to ensure they represented realistic waste collection points. OpenRouteService API was integrated using a secure key for calculating road distances and generating routes.

B. Estimated Time and Distance Calculation

```python
def calculate_time(distance):
    total_seconds = distance / (AVERAGE_SPEED * 1000 / 3600)
    hours = int(total_seconds // 3600)
    minutes = int((total_seconds % 3600) // 60)
    seconds = int(total_seconds % 60)
    return hours, minutes, seconds


def get_distance_matrix(points):
    try:
        coords = [(p[2], p[1]) for p in points]  # (Longitude, Latitude)
        matrix = ors_client.distance_matrix(
            locations=coords, profile="driving-car", metrics=["distance"]
        )
        return matrix["distances"]
    except Exception as e:
        raise ValueError(f"Error fetching distance matrix: {e}")
```

Fig 14. Estimated Time and Distance Calculation Code
Source: Author's Source Code

The program calculated the road-based distances between all waste bins using OpenRouteService's distance matrix API. A custom function computed travel time based on the total distance and an assumed vehicle speed of 15 km/h. This setup ensured that the route optimization considered real-world driving paths rather than straight-line distances.

C. Solving the Travelling Salesperson Problem

```python
def nearest_neighbor_tsp(points, distance_matrix):
    n = len(points)
    visited = [False] * n
    route = [0]
    total_distance = 0

    for _ in range(n - 1):
        current = route[-1]
        visited[current] = True
        nearest = min(
            (i for i in range(n) if not visited[i]),
            key=lambda i: distance_matrix[current][i],
        )
        total_distance += distance_matrix[current][nearest]
        route.append(nearest)

    total_distance += distance_matrix[route[-1]][route[0]]
    route.append(0)

    return route, total_distance
```

Fig 15. Travelling Salesman Problem Code
Source: Author's Source Code

The Travelling Salesperson Problem was solved using the Nearest Neighbor heuristic, which identifies the nearest unvisited point at each step. This approach, while computationally efficient, provided a practical solution for generating a Hamiltonian Circuit for the waste collection route in ITB Ganesha Campus. With the Travelling Salesman Problem Solution Approach, we can find the shortest path and find the most efficient waste collection route on ITB Ganesha campus.

D. Map Visualization and Main Program

```python
def generate_map(route_indices, total_distance, polyline):
    m = folium.Map(location=ITB_LOCATION, zoom_start=17)

    folium.PolyLine(polyline, color="blue", weight=2.5).add_to(m)

    route = [waste_points[i] for i in route_indices]
    for i, point in enumerate(route[:-1]):  # kecuali last point yang duplikat
        folium.Marker(
            location=(point[1], point[2]),
            popup=f"{i + 1}: {point[0]}",
            icon=folium.DivIcon(
                html=f
"""<div style="font-size: 12px; color: white; background: blue;
                padding: 3px; border-radius: 50%; width: 20px; height: 20px; text-
align: center;">
{i + 1}</div>"""
            ),
        ).add_to(m)

    hours, minutes, seconds = calculate_time(total_distance)
    folium.Marker(
        location=(route[0][1], route[0][2]),
        popup=f"Hamiltonian Circuit Completed<br>Total Distance: {total_distance /
1000:.2f} km<br>"
              f"Estimated Time: {hours}h {minutes}m {seconds}s",
        icon=folium.Icon(color="red", icon="info-sign"),
    ).add_to(m)

    return m


def save_and_open_map(route_indices, total_distance, polyline):
    """Save the map to an HTML file and open it in a browser."""
    m = generate_map(route_indices, total_distance, polyline)
    map_file = "bev_matdis.html"
    m.save(map_file)
    webbrowser.open(map_file)
```

Fig 16. Map Visualization Code
Source: Author's Source Code

```python
distance_matrix = get_distance_matrix(waste_points)
best_route_indices, best_distance = nearest_neighbor_tsp(waste_points,
 distance_matrix)
optimized_coords = [(waste_points[i][2], waste_points[i][1]) for i in
 best_route_indices]
polyline = get_route_data(optimized_coords)
save_and_open_map(best_route_indices, best_distance, polyline)
print("Hamiltonian Circuit generated and saved as 'bev_matdis.html'.")
```

Fig 17. Main Program Code
Source: Author's Source Code

The optimized waste collection route was visualized using Folium. Numbered markers identified each waste bin, and blue polylines represented the driving path. At the starting point, a summary of the total distance and travel time provided actionable insights for waste collection on ITB Ganesha. And then, the final implementation combined all components, from calculating distances to visualizing the optimized route. The resulting HTML map provided an intuitive interface to evaluate the waste collection plan, bridging theoretical graph concepts with the waste problem that going on in ITB Ganesha campus. The map will be saved as a file named "bev_matdis.html". The results example are illustrated in Fig. 18. And the number shown in pinpoint

shows order from where the driver start and should go.

IV. TESTING

The results of the testing are illustrated in Fig. 18 and Fig. 19, showcasing the outcomes of the map and road visualizations, respectively, using the developed source code. The program successfully generates random pinpoint locations within ITB Ganesha's campus boundaries, ensuring that all points are contained within the specified area.

The program further demonstrates its efficiency by automatically numbering the pinpoint locations based on the optimized order derived from solving the Traveling Salesperson Problem (TSP). This numbering represents the most efficient route for visiting all locations, guiding the driver through an optimal path.

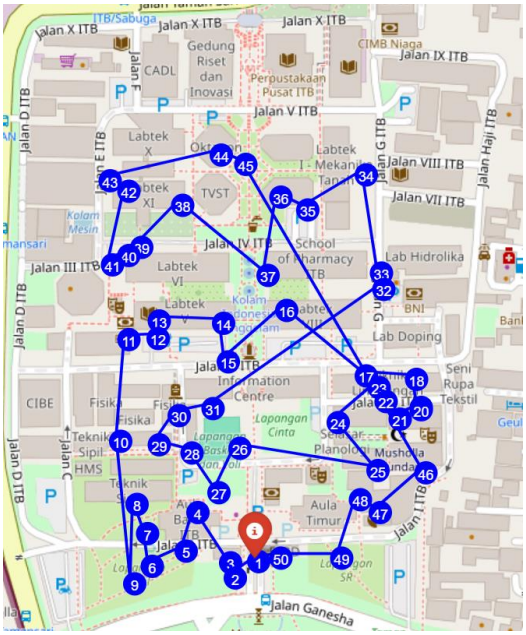

Fig 18. Testing Results of bev_matdis.html Map Visualization
Source: Author's Source Code

The route visualization in Fig. 19 clearly shows the blue-highlighted path that the driver should follow, ensuring clarity and ease of navigation. This implementation provides significant benefits for waste collection activities on ITB Ganesha campus. By visually representing both the pinpoints and the routes in an intuitive manner, the program aids drivers in planning and executing their tasks efficiently. The optimized routes minimize travel distances, reduce fuel consumption, and save time, contributing to a more sustainable and practical waste collection process within ITB Ganesha.

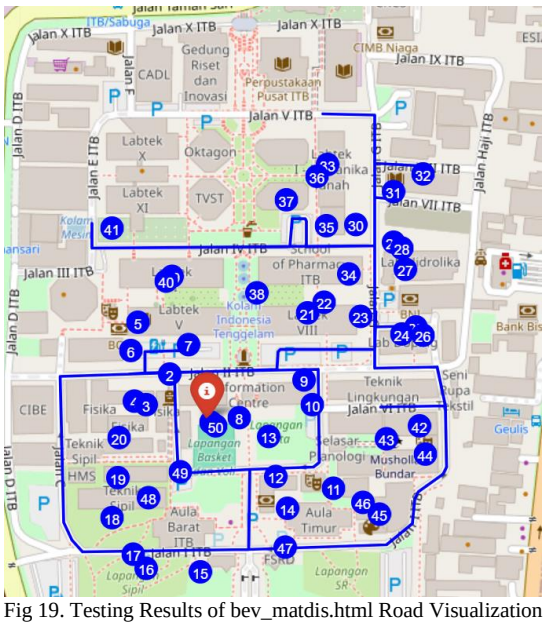

Fig 19. Testing Results of bev_matdis.html Road Visualization
Source: Author's Source Code

The results presented in Fig. 20 demonstrate the detailed distance and time information generated by the program. At the first pinpoint on the map, a red marker is displayed, which serves as the starting and ending point of the Hamiltonian Circuit. When clicked, this red pinpoint reveals a popup containing the message "Hamiltonian Circuit Completed," along with the total distance traveled in kilometers and the estimated time required to complete the route, formatted as "00h 00m 00s." This feature provides essential information for drivers, ensuring they have a clear understanding of the overall route efficiency and time required for the waste collection process.

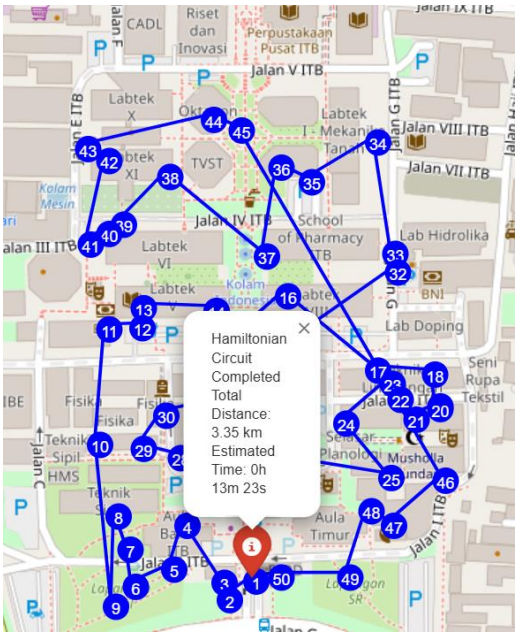

Fig 20. Testing Results of bev_matdis.html Distance and Time Information
Source: Author's Source Code

V. CONCLUSION

This study demonstrates the implementation of Hamiltonian Circuits to optimize waste collection routes within ITB Ganesha. By solving the Traveling Salesperson Problem (TSP), the program efficiently calculates the shortest route that passes through all designated waste collection points and returns to the starting point. The results include a comprehensive visualization of both the waste point locations and the optimal path, ensuring that all pinpoints and routes remain confined within ITB's campus boundaries.

The program leverages automated data processing with OpenRouteService, which generates a dynamic map showcasing numbered pinpoints and blue-highlighted routes. These features assist drivers in planning their waste collection tasks by providing clear guidance on the sequence of stops and the shortest driving path. At the starting point, the red marker includes detailed route information, such as total distance in kilometers and estimated travel time, enhancing driver efficiency and decision-making.

The implementation of this system aims to address the practical needs of ITB's waste management by reducing travel distance, fuel consumption, and time, all while ensuring sustainability and operational efficiency. Looking forward, the potential enhancements to this design could involve integrating real-time traffic data, leveraging machine learning for adaptive route optimization, or extending the system for use in larger-scale waste collection scenarios, such as city-wide operations. This solution not only benefits ITB's waste management efforts but also serves as a model for implementing efficient logistics systems in other contexts.

VI. APPENDIX

Youtube Video of Implementation of Hamiltonian Circuits on Efficient Waste Collection Route Planning on ITB Ganesha:
https://youtu.be/VCBz_rDgw88

Source Code of Implementation of Hamiltonian Circuits on Efficient Waste Collection Route Planning on ITB Ganesha:
https://github.com/bevindav/Waste-Collection-Route-ITB-Ganesha

VII. ACKNOWLEDGMENT

I would like to express my gratitude to Universe and it circumstances which have enabled me to complete this paper for IF1220 Matematika Diskrit. I personally would also like to state my biggest appreciation to the IF1220 lecturer, Ir. Rila Mandala, M.Sc., Ph.D, for teaching the course material clearly and thoroughly, making it easier for me to complete this paper without significant difficulties. Also to all of the IRK assistant who helped the process of studying Discrete Mathematics. My appreciation also goes to my mom, my sister, my friends, and who ever that had provided valuable feedback and support during the preparation of this paper.

I hope that the implementation and methods I have discussed in this paper can be developed and implemented in real-world applications and contribute to helping the society more including ITB. I personally hopes that this can help to tackle the environmental issues that Indonesia or even every world facing. Cause if not humans who maintain the earth we live, then who else?

REFERENCES

[1] Munir, R. (2024). Graf - Bagian 1. Retrieved January 4, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf.

[2] Munir, R. (2024). Graf - Bagian 2. Retrieved January 4, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf.

[3] Munir, R. (2024). Graf - Bagian 3. Retrieved January 4, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf.

[4] Twiddle, P. (n.d.). Graph Theory Notes: Hamiltonian Walks and Circuits. Retrieved January 4, 2025, from https://ptwiddle.github.io/Graph-Theory-Notes/s_walks_hamiltonian.html.

[5] Institut Teknologi Bandung (ITB). (2023). Studium Generale ITB: Pengelolaan Sampah Berkelanjutan untuk Masa Depan yang Lebih Bersih dan Sehat. Retrieved January 5, 2025, from https://itb.ac.id/berita/studium-generale-itb-pengelolaan-sampah-berkelanjutan-untuk-masa-depan-yang-lebih-bersih-dan-sehat/61356.

PERNYATAAN

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

Bandung, 6 Januari 2025

Bevinda Vivian 13523120