Exploring Optimal Path Solutions in Mini Metro Game Through Graph Analysis and Game Strategy

Barru Adi Utomo - 13523101[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]barru.adi@gmail.com, 13523101@std.stei.itb.ac.id

***Abstract*— Mini Metro, a dynamic subway simulation game, mirrors real-world transportation challenges by requiring players to connect stations, manage passenger flow, and adapt to evolving demands. This study applies graph theory and optimization algorithms to model stations as nodes and routes as edges, optimizing network performance. Strategies like prioritizing loops over linear paths and proximity-based connections were evaluated, highlighting their effectiveness in reducing travel time and enhancing accessibility. Future work should focus on adaptive algorithms and predictive modeling to improve optimization and scalability. This research bridges theoretical graph algorithms and practical gameplay strategies.**

***Keywords*—Mini Metro, Graph theory, Railway network, Algorithm optimization.**

I. Introduction

In modern game design, optimization challenges often mirror real-world problems, particularly in transportation and logistics. The game Mini Metro exemplifies this by simulating the creation and management of a subway system. Players must connect stations, adapt to dynamically emerging demands, and ensure efficient passenger transportation under resource constraints.

Fig 1. Mini Metro Game

The core of Mini Metro's mechanics aligns closely with graph theory, where stations can be modeled as nodes, and routes as edges. Solving the problem of optimal pathfinding in such a system is not only critical for achieving high scores in the game but also provides a microcosm of real-world transportation challenges. Traditional optimization algorithms, such as Dijkstra's or Minimum Spanning Tree, offer potential solutions to these challenges by identifying efficient paths and minimizing congestion.

Fig 2. Mini Metro Gameplay

This study explores the application of graph analysis and strategic gameplay to optimize routes in Mini Metro. By simulating gameplay scenarios and applying graph-based algorithms, this research aims to uncover strategies that enhance network efficiency and provide players with actionable insights. Furthermore, it delves into how these strategies adapt to dynamic changes, such as the emergence of new stations or increased passenger load, replicating the evolving complexities of real-world networks.

Through this exploration, we bridge the gap between theoretical graph algorithms and practical game strategies, contributing both to game optimization and broader applications in dynamic system management.

II. Graph

A graph is a mathematical structure used to represent relationships or connections between objects. It consists of vertices (or nodes), which represent the objects, and edges (or links), which represent the connections between them [1].

A. Graph Terminology

a. Adjacent

Two vertices in a graph are considered adjacent if there is an edge connecting them. This indicates a direct relationship between the two nodes, such as a direct route between stations in Mini Metro. For instance, if Station A is directly connected to Station B, these two stations are adjacent in the graph representation.

b. Incidency

The relationship between a vertex and an edge is known as incidency. A vertex is incident to an edge if it is one of the edge's endpoints. In the Mini Metro network, a station is incident to all the routes that pass through it. For example, if a route connects Station A to Station B, both stations are incident to the edge representing that route.

c. Isolated Vertex

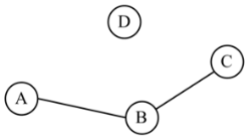

Fig 3. Isolated Vertex

A vertex with no edges connecting it to any other vertex is called an isolated vertex. Such vertices represent stations in the Mini Metro network that are not part of any route, making them inaccessible to passengers. Identifying and minimizing isolated vertices is crucial for ensuring a fully connected transportation network.

d. Degree

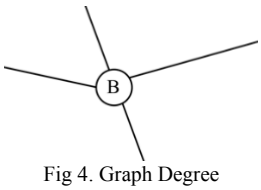

Fig 4. Graph Degree

The degree of a vertex is defined as the number of edges incident to it, representing the number of direct connections a station has to other stations. In directed graphs, the degree can be further classified into in-degree (the number of incoming connections) and out-degree (the number of outgoing connections). For example, a station connected to three other stations would have a degree of three.

e. Path

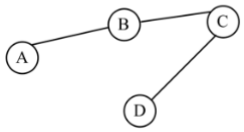

Fig 5. Graph Path

A path in a graph is a sequence of vertices connected by edges, where each edge is traversed only once [2]. In Mini Metro, a path corresponds to a passenger's journey from one station to another, passing through a series of connected routes. Paths are fundamental for understanding the movement of passengers and optimizing their travel times.

f. Circuit

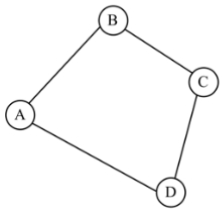

Fig 6. Graph Circuit

A circuit is a closed path that starts and ends at the same vertex, with no edge repeated [2]. Circuits are commonly used in transportation systems to design loop routes, ensuring efficient movement across a network. In Mini Metro, a circuit represent a line that allows trains to travel around a set of stations without the need to reverse direction.

g. Weighted Graph

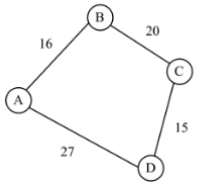

Fig 7. Weighted Graph

A weighted graph assigns numerical values (weights) to its edges, often representing costs, distances, or travel times [3]. In Mini Metro, weights can be used to model factors such as travel time between stations or congestion levels on a route. Weighted graphs are essential for implementing optimization algorithms to find the most efficient paths.

B. Euler Path and Circuit

An Euler Path is a path in a graph that traverses each edge exactly once. It does not need to start and end at the same vertex. For a graph to have an Euler track all edges in the graph must be connected and exactly two vertices in the graph have an odd degree [4].

C. Graph Coloring

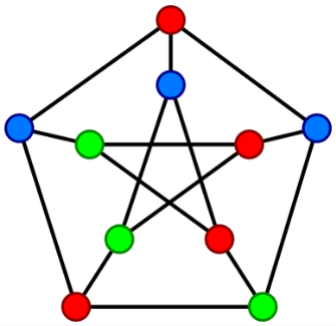

Fig 8. Graph Coloring

Graph coloring is the assignment of colors to the vertices of a graph such that no two adjacent vertices share the same color. This is known as vertex coloring. The minimum number of colors required to achieve this is called the chromatic number of the graph [5]. In Mini Metro, graph coloring is used to

differenciate station shapes so that it optimize the train passanger.

III. Algorithm

An algorithm is a finite set of well-defined instructions used to solve a specific problem or perform a computation. In the context of graph theory, algorithms are essential for processing, analyzing, and optimizing graph structures to achieve desired outcomes, such as finding shortest paths, minimizing costs, or determining connectivity.

A. Djikstra

Dijkstra's Algorithm is a well-established method for solving the shortest path problem in weighted graphs. Originally proposed by Edsger W. Dijkstra in 1956, the algorithm efficiently determines the shortest path from a single source node to all other nodes within a graph [6]. Its applications span diverse fields, including network routing, transportation planning, and resource optimization.

For each unvisited neighbor v of u, calculate the alternative distance via u as:

$$\text{distance}[v] = \min(\text{distance}[v], \text{distance}[u] + \text{weight}(u,v)) \quad (1)$$

If this alternative distance is smaller than the current distance, update distance[v] and update its position in the priority queue.

IV. Research Methodology

A. Problem Definition

The Mini Metro game presents a dynamic optimization problem, where stations appear unpredictably, and players must connect them efficiently while managing passenger flow and limited resources. This study seeks to address the following core challenges:

- Designing optimal routes to minimize travel time and maximize passenger satisfaction.
- Adapting to dynamic changes such as new station appearances and varying passenger demand.
- Balancing the trade-offs between shortest paths, resource allocation, and network efficiency.

To address these challenges, the research models the game using graph theory, where stations represent nodes, and routes represent weighted edges. By analyzing the problem with graph-based algorithms, the study aims to uncover strategies that improve gameplay performance and offer broader insights into dynamic system optimization.

Python libraries that are being used are NetworkX, PyGame, Random, and Math

```
# Import library
import pygame
import networkx as nx
import random
import math
```

B. Data Collection

Data is collected directly from Mini Metro gameplay and replicate in python script. Gameplay data includes station locations, passenger destinations, and route configurations. For instance, random coordinates from the gameplay are projected into the replica, and connections are established based on certain algorithm. This dataset serves as the foundation for creating the graph representation of the game's network and testing optimization strategies.

```
# Variables

WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)

station_type = [
    'square', 'circle', 'triangle',
    'square', 'circle', 'triangle',
    'square', 'circle', 'triangle',
    'square', 'circle', 'triangle',
    'unique'
]

node_positions = {
    'circle': {},
    'square': {},
    'triangle': {},
    'unique': {}
}

global_node_id = 1
global_circle_count = 0
global_square_count = 0
global_triangle_count = 0
global_unique_count = 0
```

C. Graph Representation

The collected data is transformed into a graph representation using NetworkX, a Python library for graph manipulation.

```
# Initialization
pygame.init()
screen = pygame.display.set_mode((800, 600))
graph = nx.Graph()
```

Nodes represent stations, each assigned attributes such as location (coordinates) and type (e.g., circle, square, triangle, and unique).

Edges represent connections between stations, weighted by distance and game strategy. For example, a connection between two stations is assigned a weight proportional to the Euclidean distance between their coordinates. This graph serves as the input for various algorithms, enabling the study to simulate and analyze the Mini Metro network efficiently.

D. Algorithm and Game Strategy

There are a lot of game strategy. Game strategy will affect algorithm that is going to be used. This game strategy is taken from Hamy's blog, where it talks about with this subject "Mini

Metro - 5 Beginner Tips to Reach Top 10% on the Leaderboard".

There are four main strategy that can be applied to this algorithm:

- Avoid duplicate station
- Prefer loops to point-to-point lines
- Prefer line connections on unique stations
  There are generally 4 types of stations: circle, triangle, square, and unique shapes like star, plus oval, etc.
- Connect loops together with express trains

With that, this study requires a specific algorithm, and this is the algorithm that is used.

```
# Algorithm

def calculate_algorithm():
    lengths = {}
    for node1 in graph.nodes:
        for node2 in graph.nodes:
            if node1 != node2:
                pos1 =
graph.nodes[node1]['pos']
                pos2 =
graph.nodes[node2]['pos']
                length = math.sqrt((pos1[0]
- pos2[0]) ** 2 + (pos1[1] - pos2[1]) ** 2)
                lengths[(node1, node2)] =
length

    circuits = []
    used_nodes = set()
    while True:
        circuit = []
        for shape in ['circle', 'square',
'triangle', 'unique']:
            nodes = [node for node in
node_positions[shape].keys() if node not in
used_nodes]
            if nodes:
                node = nodes[0]
                circuit.append(node)
                used_nodes.add(node)
        if len(circuit) >= 3:
            circuits.append(circuit)
        else:
            break

    for shape in ['circle', 'square',
'triangle']:
        nodes = [node for node in
node_positions[shape].keys() if node not in
used_nodes]
        if nodes:
            node = nodes[0]
            circuit.append(node)
            used_nodes.add(node)
    if len(circuit) >= 3:
        circuits.append(circuit)

    all_nodes = list(graph.nodes)
    for i in range(len(all_nodes)):
        node1 = all_nodes[i]
        node2 = all_nodes[(i + 1) %
len(all_nodes)]
        pos1 = graph.nodes[node1]['pos']
        pos2 = graph.nodes[node2]['pos']
        if node1 not in used_nodes or node2
not in used_nodes or
graph.nodes[node1]['shape'] == 'unique' or
graph.nodes[node2]['shape'] == 'unique':
            pygame.draw.line(screen, WHITE,
pos1, pos2, 2)
            graph.add_edge(node1, node2)
```

* full algorithm is in github repository in Chapter VII

E. Visualization

Visualization using PyGame to minimize the randomness of the station generated and the user can adjust where the station will appear. The algorithm is also called each time the station appeared.

```
# Visualize


while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        elif event.type ==
pygame.MOUSEBUTTONDOWN:
            if event.button == 1:
                add_station(event.pos)
                draw_graph()
                calculate_algorithm()
    pygame.display.flip()
```

V. EXPERIMENT

Using the game methodology as mentionted in Chapter IV, the program will output a blank canvas and as the user click the canvas, random shape will appear and the algorithm will try to find a path to solve the current nodes.

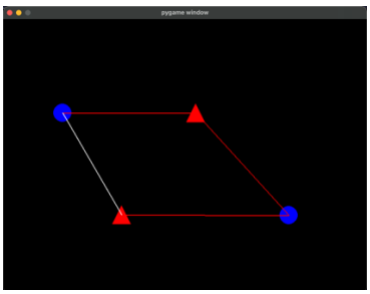

Fig 9. First Experiment

Fig 9 shows the first game strategy, that is to avoid duplicate station. The order of the circuit is circle, rectangle, circle, rectangle. Fig 9 also shows the second game strategy, to prefer loops to point-to-point lines. Loops are better than straight line becausse loops having less and equal time to reach station respectively.

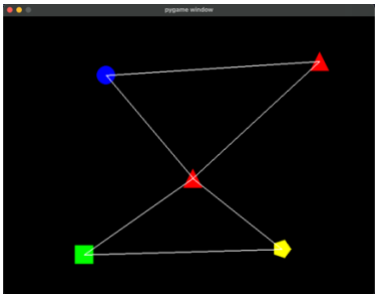

Fig 10. Second Experiment

Fig 10 shows the another example of the game strategy being implemented in the algorithm, even though it is better to use the unique shape to connect between loops, but the distances are much longer to reach the unique shape as the line connections.

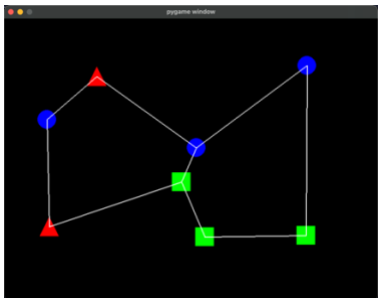

Fig 11. Third Experiment

Fig 11 shows that a more dynamic the algorithm can be. There are two adjacent that are so close in the middle. The algorithm decided that it is more effective to connect two of the close one.

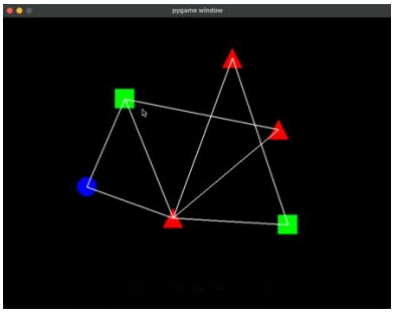

Fig 12. Fourth Experiment

As problems described in Chapter IV, the experiment designed to find the design the optimal route to minimize travel time and maximize passenger satisfaction, adapting to dynamic changes such as new station appearances and varying passenger demand, and balancing the trade-offs between shortest paths, resource allocation, and network efficiency.

VI. Conclusion

This study explores strategies and algorithmic approaches for optimizing the Mini Metro game network, focusing on dynamic graph-based solutions. The findings indicate that prioritizing unique station connections and incorporating loops into the network design can significantly enhance efficiency by minimizing travel time and ensuring better accessibility. Loops, in particular, demonstrate their superiority over point-to-point connections, offering more robust and adaptable networks. Additionally, the algorithm shows promise in optimizing connections by prioritizing proximity, effectively reducing travel distances and improving connectivity.

However, the algorithm faces significant limitations in handling the inherent randomness of the game. It does not account for unpredictable station placement, varying passenger demands, or the dynamic rewards and upgrades that emerge during gameplay, such as additional rail lines, new trains, or station enhancements. These factors add complexity and require advanced computation and more sophisticated decision-making models. Moreover, game mechanics such as bridge placement and congestion management further challenge the algorithm's adaptability and scalability.

Future research should address these limitations by developing adaptive algorithms capable of dynamically adjusting to the game's randomness and evolving network structure. Incorporating multi-objective optimization techniques, predictive modeling for station and passenger behavior, and real-time decision-making frameworks will enhance the algorithm's performance. Additionally, testing scalability on larger networks and integrating insights from gameplay dynamics can ensure practical applicability and improved efficiency. By tackling these challenges, future work can significantly advance the effectiveness of graph-based strategies in dynamic systems like Mini Metro.

VII. Appendix

1. Source code:
   https://github.com/barruadi/mini-metro-graph

2. Explanation Video:
   https://youtu.be/Yenf0Ahhv6Q

VIII. Acknowledgment

I would like to express my gratitude to the individuals and institution whose support and contributions have been instrumental in the completion of this research:

1. God Almighty, thanks to His grace and guidance, this paper can be completed.
2. Both parents who have been supportive for this research.
3. Dr. Ir. Rinaldi Munir M.T. as the lecturer of the IF1220 Discrete Mathematics.
4. All of my friends, especially Azfa Radhiyya Hakim and Rafif Farras who also support me in this research.

REFERENCES

[1] P. Zhang and G. Chartrand, "Introduction to graph theory," *Tata McGraw-Hill*, vol. 2, pp. 2–1, 2006.

[2] J. A. Bondy, "Basic graph theory: paths and circuits," *Handbook of combinatorics*, pp. 1–2, 1995.

[3] J. A. Bondy and G. Fan, "Cycles in weighted graphs," *Combinatorica*, vol. 11, pp. 191–205, 1991, Springer.

[4] N. O. Acosta and A. I. Tomescu, "Simplicity in Eulerian circuits: Uniqueness and safety," *Information Processing Letters*, vol. 183, p. 106421, 2024, Elsevier.

[5] P. Jovanović, N. Pavlović, I. Belošević, and S. Milinković, "Graph coloring-based approach for railway station design analysis and capacity determination," *European Journal of Operational Research*, vol. 287, no. 1, pp. 348–360, 2020, Elsevier.

[6] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," in *SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man, and Cybernetics: Cybernetics evolving to systems, humans, organizations, and their complex interactions (Cat. No. 0)*, vol. 3, pp. 2316–2320, 2000, IEEE.

STATEMENT

I hereby declare that this paper is my own writing, not an adaptation, or translation of someone else's paper, and not plagiarized.

Bandung, 8 January 2024

Barru Adi Utomo
13523101