Quantifying Traffic Decongestion Using Public Transport in *Cities: Skylines II*

Ghiffari Arya Adhitya - 135250461[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: [1]ghiffariarya24407@gmail.com, [1]13525046@std.stei.itb.ac.id

Abstract—Managing urban traffic is a challenging task in modern city planning and management. Utilizing *Cities: Skylines II*'s simulation, this paper maps commuter hotspots as nodes (N) within a directed network graph, where edges (E) are defined as commuter travel times. This paper calculates the efficiency of these network graphs using the Floyd-Warshall all-pairs shortest path algorithm to analyze the global travel cost difference before and after public transport implementation. Comparative matrix analysis shows that public transport implementation provides a 18.13 % reduction in global travel cost. The comparative matrix analysis proves that optimizing transit around key commuter hotspots yields results for traffic decongestion.

Keywords—Cities: Skylines II, Traffic Decongestion, Graph Theory, Floyd-Warshall Algorithm, All-Pairs Shortest Path

I. Introduction

Managing urban traffic congestion is a critical task in modern civil engineering and network optimization. As metropolitan areas scale, the reliance on car-centric infrastructure frequently introduces localized bottlenecks, which degrades the efficiency of global transit networks. Traditional traffic management techniques often rely on road expansions, which may not be possible due to fiscal restraints and social constraints. Consequently, contemporary urban planning increasingly leverages advanced computational modeling and graph theory to analyze alternative transportation solutions before field implementation.

To analyze these complex networks under realistic conditions, a macro-simulation environment such as *Cities: Skylines II* offers a valuable testing ground. *Cities: Skylines II* simulates commuter pathfinding, variable vehicle speed and acceleration, and intersection delays under variable peak-load conditions.

This paper analyzes the efficacy of public transport implementation to reduce traffic congestion using the Floyd-Warshall algorithm. The matrices are modelled using a directed network graph, with commuter hotspots as vertices and weighted edges as commuter travel times between hotspots. Efficacy is calculated by comparing the global transit cost of the matrix before and after public transport implementation. Therefore, we can quantify the efficiency of public transport implementation for our transit network.

II. Theoretical Background

A. Graph

Graphs are structures that consist of vertices (objects) and edges that represent connections between vertices in a discrete manner.

Graphs can be defined as G = (V, E) with V being a non-empty set of vertices, example: V = $\{v_1, v_2, \dots, v_n\}$. E is defined as a set of edges that each connects a pair of vertices, example: E = $\{e_1, e_2, \dots, e_n\}$. Two vertices can have more than one edge that connects to one another, these are called multiple edges or parallel edges. An edge can also connect to itself, this is called a loop (See Image 1, e_8 in G_3).

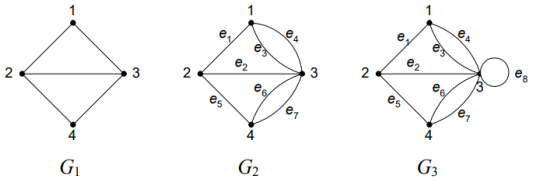


Image 1 (a) Simple Graph, (b) Multi-Graph, and (c) Pseudo-Graph
Source: [1]

Edges and vertices in graphs can be given labels, these can signify weights, directions, or just simple names. Therefore, labels don't have to be numbers or integers, they can be text, and/or whatever label the graph needs. An example of a graph with labels are electrical circuits (The labels are the symbols in the edges that signify capacitors, resistors, etc).

There are multiple types of graphs, differentiated by their properties. These properties are whether they are allowed to have parallel graphs, loops, and if they have directional edges.

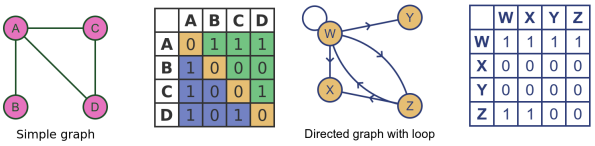
Table 1 Types of graphs, Source: [1]

Type	Directional Edges?	Allowed Parallels?	Allowed loops?
Simple Graph	No	No	No
Multi Graph	No	Yes	No
Pseudo Graph	No	Yes	Yes
Directional Graph	Yes	No	Yes
Multi-Directional Graph	Yes	Yes	Yes

Graphs can also be represented as a matrix which makes it easier to analyze and manipulate mathematically. Following this are some of the ways graphs can be represented as matrices.

1. Adjacency Matrix

An adjacency matrix can be defined as A = [a_{ij}], where i and j are indexes in the matrix that represent vertices. Therefore, a_{ij} is defined as the edge that connects between vertex i and vertex j.

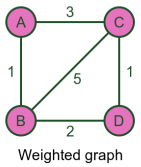


	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	1	0	0	1
D	1	0	1	0

	W	X	Y	Z
W	1	1	1	1
X	0	0	0	0
Y	0	0	0	0
Z	1	1	0	0

Image 2 Adjacency Matrix, source: [2]

Weighted graphs can also be represented as an adjacency matrix. This means that a_{ij} isn't limited to 1s or 0s, they can be any number that represents the weight of the connection.



	A	B	C	D
A	0	1	3	0
B	1	0	5	2
C	3	5	0	1
D	0	2	1	0

Image 3 Weighted Adjacency Matrix, source: [2]

In image 3's case, vertices that have no connections are represented as having a value of 0. This makes sense if we are representing distance or road connections, because a distance of 0 doesn't make sense. If we do need to set a weight of 0 in our graphs, we can represent not having a connection as a different number, say a value of -1.

2. Incidency Matrix

This matrix is slightly different from an adjacency matrix, mainly because an incidency matrix has its rows as the vertices and its columns are the edges of the graph. Edges that connect to vertices are represented as 1 and -1 (for directional graphs), and 0 if they don't connect to that vertex.

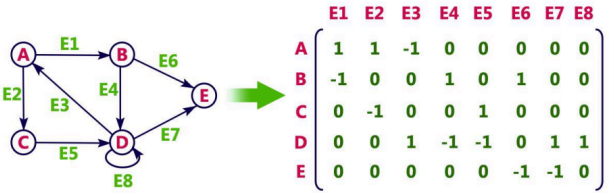


Image 4 Incidency Matrix, source: [3]

B. *The Floyd-Warshall Algorithm*

The Floyd-Warshall algorithm is a dynamic programming algorithm for finding the shortest path in a directional weighted graph with positive and/or negative edge weights. The algorithm checks all vertices (i, j) then checks if an intermediate path (k) results in a shorter path. However, this algorithm doesn't work with negative cycles (the sum of the edges in a cycle are negative).

```
Floyd-Warshall(W)
    n = W.rows
    D^(0) = W
    for k = 1 to n
        let D^(k) = (d_ij^(k)) be a new n × n matrix
        for i = 1 to n
            for j = 1 to n
                d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1)+d_kj^(k-1))
    return D^(n)
```

Image 5 Floyd-Warshall Algorithm Pseudocode

Source: [4]

The Floyd-Warshall algorithm first takes in the argument W, which is the initial weighted adjacency matrix. The algorithm then sets the base of the recurrence as W, the initial adjacency matrix. It then loops to find the shortest path between index i and j by checking if a detour through vertex k results in a shorter path.

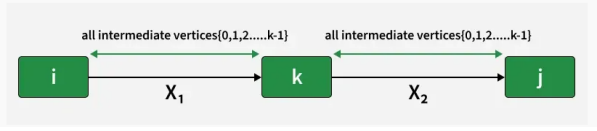


Image 6 Optimal Substructure Property in Floyd-Warshall Algorithm, source: [5]

It's important to note that the Floyd-Warshall algorithm has an algorithmic complexity of $\Theta(n^3)$, with n being the number of vertices. This is due to the algorithm using three nested loops (for k, for i, for j) to search across all elements of n and checks all possible pairs to find the most optimal path from vertex i to vertex j.

C. *Public Transportation*

Public Transportation can be defined as a system of vehicles such as buses and trains that operate at regular times on fixed routes that are used by the public[6]. However, in the last few decades, public transportation is largely viewed as a dated way of transportation in comparison to the private car. The shift from mass transit such as trains to large intercontinental highways has made mass public transportation into mismanagement and budget cuts. This creates the perception that public transportation is mostly an outdated system.

However, this doesn't have to be the case. Proper management and integration with digital systems, can make public transportation a much more convenient way of transport than even the private car. This is the case in Europe, Japan, and other countries. Even in Indonesia, the digital infrastructure allows for buying tickets for trains and when you get to the train station, you just need to do a facial scan to get in. No need to wait at ticketing stations anymore.

Public Transportation is generally considered a significantly more efficient way of transportation compared to a car. This is true in terms of energy efficiency (Joules per meter per person), it's also generally more environmentally friendly, the exception being planes. This is simply because of the sheer amount of people, systems such as trams and metro systems, can move at a given moment. This also indirectly reduces the amount of traffic congestion a city has, because each person riding public transport is another person that isn't driving a car. This means less gridlocks, less waiting at intersections, less queues at tollways, etc.

Public transport also provides more inclusivity for people who can't use a private car. This can be due to being handicapped, elderly, or too young to drive. Some people also can't afford to own a car, this often falls under the previously said categories. If these people need to have the mobility essential for their subsistence and satisfaction of their lives, then public transport is a necessity.

Typical capacities of urban mass transportation modes

vehicle type	guideway type	persons per vehicle		"train" length	"trains" per hour		average speed	passengers per hour	
		seated	crowded		low	high		low	high
auto	streets	1.2	3	1	450	900	20	540	2,700
auto	freeways	1.2	3	1	900	1,800	30	1,080	5,400
bus	streets	50	80	1	1	60	8	50	4,800
bus	separate	50	80	1	1	30	20	50	2,400
light rail	streets	80	100	3	6	30	10	1,440	9,000
light rail	separate	90	120	4	4	30	25	1,440	14,400
heavy rail	separate	100	120	8	4	40	30	3,200	38,400
commuter rail	separate	100	150	10	1	12	35	1,000	18,000

Image 7 Typical Capacities of Urban Mass Transportation Modes

Source: [7]

III. Algorithmic Formulation

A. *Network Modeling and Base Case Initialization*

Let the urban road network in *Cities: Skylines II* be modelled as a directed weighted graph G = (V, E), where V = {v_1 , v_2 , ... , v_n} represents the set of n commuter hotspots and E represents the set of connecting roads. Each edge (i, j) ∈ E is assigned to a dynamic weight $W_{ij} \in R^+$ corresponding to travel times between hotspots (vertex i and j). Therefore, the weight matrix of n x n size can be initialized as:

$$w_{ij} = \begin{cases} 0, & \text{if } i = j \\ \text{travel time}, & \text{if } (i,j) \in E \\ \infty, & \text{if } (i,j) \notin E \end{cases}$$

Image 8 Weight Matrix Definition

B. *Inductive Step and Recurrence Relation*

The algorithm solves the network optimization problem dynamically by first constructing a sequence of matrices $D^{(1)}$, $D^{(2)}$, …, $D^{(n)}$. Each structural state of $D^{(k)} = (D_{ij}^{(K)})$ defines the shortest path from vertex i to vertex j, while ensuring that any intermediate vertices are strictly elements from the subset {v_1 , v_2 , ... , v_n}.

For each inductive step, the transition from matrix $D^{(k-1)}$ to $D^{(k)}$ is controlled by the following recurrence relation for all i, j ∈ {*1, 2, ..., n*}:

$$D_{ij}^{(k)} = \min(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)})$$

The left operand of the min function represents the existing optimal path that doesn't use vertex k as an intermediary node (i→j). The right operand represents the new path weight using vertex k as an intermediary node (i→k→j). The function then sets the lowest value of the two operands as the shortest path between vertex i and vertex j.

C. *Final State*

Upon finishing the final loop where k = n, the algorithm results in the final matrix $D^{(n)}$. Every cell value $D_{ij}^{(n)}$ in this matrix is mathematically the shortest, most optimal path from vertex i to vertex j.

IV. Case Study

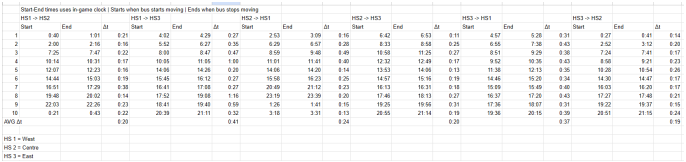
A. Data Collection

This paper will use a fictional city in *Cities: Skylines II*, named Kimiko City as its subject. Initial commuter hotspot detection will be using the *Transit Hotspots* mod made by Siont. This mod records observed citizen trips and highlights high-demand hotspots. This tool is useful for identifying regions that may need support systems to fulfill public transport demand.

Image 9 Kimiko Transit Hotspots with more than 550x demand

From the preceding image, we can see that Kimiko has 2 major areas in need of transport options. However, due to the scale of the eastern (right) area, splitting it into two separate hotspots is an ideal solution. Therefore, Kimiko city has 3 separate hotspots.

To calculate the travel times between hotspots, a singular bus with the maximum ticket price (significantly reduces demand) will be used. The calculation will use the in-game clock, not real time, so that the time isn't affected by the game's often oscillating simulation speed. To ensure accurate calculations, the Δt will be calculated by the End_Time - Start_Time of the bus route. Start_Time is the time of the in game clock when the bus starts moving from a bus stop. End_Time is the in-game clock time of when the bus arrives at



its destination. This process will then be repeated 10 times to get the average Δt from that hotspot to its destination hotspot. The following is the time data before public transport implementation.

Image 10 Kimiko Transit Times Before

It's interesting to note here that some of the values, especially around 06:00-10:00 and 17:00-20:00, have a higher Δt than the rest of the values. This is because of *Cities: Skylines II* rush hour mechanic where traffic volumes in this time range becomes significantly worse due to the amount of people entering and leaving work at the same time frame.

Public Transport implementation will use a main bus transit line in Kimiko's main road, connected to a transit hub near a train station. This train station will facilitate transportation exiting or entering the city. The bus line's ticket price will also be set to free to increase bus ridership.

Image 11 (Left) Transit Hub Next to Train Station
Image 12 (right) Kimiko Main Bus Line

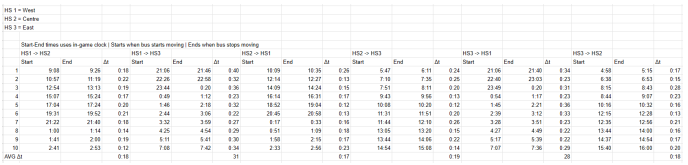


Image 13 Kimiko Transit Times After

After implementing the transport system, travel times have reduced a substantial amount compared to before implementation.

B. Data Analysis

Analyzing data will be using the Floyd-Warshall Algorithm, implemented in python. This program will be used to optimize the directional weighted graph before comparing its final total network cost. The following is the code implementation.

```
import numpy as np

## Floyd-Warshall Shortest Path Algorithm
def floyd_warshall(matrix):

    distance = np.copy(matrix)
    vertex_count = distance.shape[0]

    for k in range(vertex_count):
        for i in range(vertex_count):
            for j in range(vertex_count):
                if distance[i, k] + distance[k, j] < distance[i, j]:
                    distance[i, j] = distance[i, k] + distance[k, j]

    return distance
```

Using the data from section A, we can calculate the total network cost of both systems and compare their values. The following is the data, represented as a directional weighted matrix.

Before Floyd-Warshall			
Before Implementation			
Vertex	W	C	E
W	0	20	41
C	24	0	20
E	37	19	0
After Implementation			
Vertex	W	C	E
W	0	18	31
C	17	0	19
E	28	18	0

After Floyd-Warshall			
Before Implementation			
Vertex	W	C	E
W	0	20	40
C	24	0	20
E	37	19	0
After Implementation			
Vertex	W	C	E
W	0	18	31
C	17	0	19
E	28	18	0

Image 14 (left) Directional Weighted Matrix Before Algorithm Implementation (W = West, C = Central, E = East)

Image 15 (right) Directional Weighted Matrix After Algorithm Implementation (W = West, C = Central, E = East)

After implementing the Floyd-Warshall algorithm, this results in a total cost of 160 minutes for the car-centric design. Meanwhile, the public transit design has a total network cost of 131 minutes.

V. Results and Discussion

A. Global Network Cost Reduction

To evaluate the efficiency of the public transit implementation, we have to compare the total network cost of the final matrix ($D^{(n)}$) with the baseline matrix ($D^{(0)}$). The total network cost (C) is defined as the summation of all optimal travel times across all evaluated vertex pairs.

$$C = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}$$

From the results gathered in Section IV.B, the baseline network cost ($C_{baseline}$) is 160 minutes. Meanwhile the public transport network cost ($C_{optimized}$) is 131 minutes. This results in a difference in total network travel cost of 29 minutes. This represents a 18.13% efficiency optimization in commuter throughput throughout the transit network.

B. Discussion

The main difference seen between before and after public transit implementation are at rush hours. With public transit, traffic during rush hours is much more mild than compared to when there wasn't any public transport. Before, travel times could reach above 1 hour (in-game time) in peak rush hour. With the addition of public transport, traffic volumes reduced significantly. However, this also caused an increase in pedestrian traffic, which caused some intersections to become significantly busier. This caused a slight increase in delays in intersection wait times. Although this increased intersection delays, the decrease in overall traffic volume outweighed the drawbacks and resulted in the 18.13% increase in network efficiency.

Recently the developers of *Cities: Skylines II*, *Iceflake Studios*, have announced that they will be reworking their traffic modeling to more accurately reflect real life traffic. Therefore it's recommended that the results of this paper be revisited after these changes have been implemented into the game.

VI. Conclusion

This paper demonstrates the application of the Floyd-Warshall all-pairs shortest path algorithm to optimize urban traffic simulated within the *Cities: Skylines II* simulation. By modeling a directional weighted graph using hotspots as vertices and travel times as edges, the efficiency of public transit implementation could be quantified using said algorithm.

The implementation of public transit systems altered the baseline graph by creating reductions in edge weights. The new system achieved a definite decrease in total network travel cost from 160 minutes to 131 minutes. This difference of 29 minutes resulted in an increase of transit network efficiency of 18.13%.

Future additions to this research could utilize laplacian graphs to perform spectral graph clustering. This would allow automatic partitioning of the routing system based on organic bottleneck boundaries. This would mean increasing the computational efficiency and allows for the modeling of ever expanding megacities.

Acknowledgements

The author would like to thank, Prof. Dr. Ir. Rinaldi, M.T, for their valuable insight and guidance. The author would also like to convey their thanks and gratitude towards both of their parents and friends for their support and encouragement for the author. The author would also like to thank the open source community for providing reference material for this paper.

References

[1] R. Munir, "Graf (Bagian 1)," lecture notes for IF1220 Matematika Diskrit, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf [Accessed: Jun. 19, 2026].

[2] M. McBride, "Adjacency matrices," *GraphicMaths*, Aug. 17, 2025. [Online]. Available: https://graphicmaths.com/computer-science/graph-theory/adjacency-matrices/ [Accessed: Jun. 19, 2026].

[3] R. Munir, "Graf (Bagian 2)," lecture notes for IF1220 Matematika Diskrit, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf [Accessed: Jun. 19, 2026].

[4] "Floyd-Warshall algorithm pseudocode," lecture handout for CMSC351: Algorithms, Dept. Computer Science, Univ. Maryland, College Park, MD, USA, Fall 2021. [Online]. Available: https://www.cs.umd.edu/class/fall2021/cmsc351-0301/files/floydWarshall.pdf [Accessed: Jun. 19, 2026].

[5] "Floyd Warshall algorithm," *GeeksforGeeks*, Feb. 25, 2026. [Online]. Available: https://www.geeksforgeeks.org/dsa/floyd-warshall-algorithm-dp-16/ [Accessed: Jun. 19, 2026].

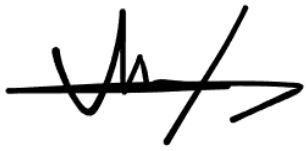
[6] "Public transport," *Cambridge Advanced Learner's Dictionary & Thesaurus*, Cambridge University Press. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/public-transport [Accessed: Jun. 19, 2026].

[7] J. L. Schofer, "Mass transit: The benefits of urban mass transit," *Encyclopedia Britannica*, Jun. 11, 2026. [Online]. Available: https://www.britannica.com/topic/mass-transit/The-benefits-of-urban-mass-transit. [Accessed: Jun. 19, 2026].

DECLARATION

I hereby declare that this paper is my own work, not an adaptation or translation of another person's paper, and is not plagiarized.

Bandung, June 19th, 2026



Ghiffari Arya Adhitya 13525046