Min-Transfers Algorithm for Temporal Paths Preservation in Public Transportation Networks

Ghulam Mustafa and Syeda Wania
Team 46

April 27, 2025

Abstract

This report presents the implementation and evaluation of the Min-Transfers Algorithm, which optimizes public transportation systems by minimizing vehicle transfers. The algorithm operates on temporal graphs representing transport networks and converts them into Temporal Paths Preservation (TPP) graphs to find the minimum transfer path. The report discusses the algorithm's design, implementation, results, and future work.

1 Background and Motivation

1.1 Context

Public transportation is a critical component of urban mobility, providing an efficient alternative to private vehicles. However, commuters often face inconveniences such as frequent vehicle transfers, leading to longer travel times and a less optimal experience. Minimizing the number of transfers can significantly improve the commuter experience, reduce overall travel time, and increase the efficiency of transport systems.

1.2 Problem Statement

The goal of this project is to develop an algorithm that minimizes the number of vehicle transfers required to travel between two stations in a temporal transportation network, where each edge (trip) has a departure time, duration, and vehicle ID. This is important for improving the efficiency of public transportation systems and enhancing commuter satisfaction.

1.3 Motivation

- **Time Efficiency**: Minimizing transfers reduces travel time by eliminating unnecessary waiting times between vehicles.
- **Convenience**: Fewer transfers mean less hassle, especially for people who are unfamiliar with the network or have mobility issues.
- **Environmental Impact**: Reducing transfer times can help lower emissions and reduce the overall carbon footprint of the transportation system.

2 Algorithm Overview

2.1 Original Algorithm

The Min-Transfers Algorithm operates on a temporal graph, where each trip is represented by an edge with a departure time, duration, and vehicle ID. The algorithm aims to minimize the number of transfers required to reach a destination while maintaining temporal and vehicle constraints.

2.2 Main Idea

- **Temporal Graph to TPP Graph**: The algorithm first converts the temporal graph into a Temporal Paths Preservation (TPP) graph. Each trip becomes a node, and valid transitions between trips (based on temporal constraints) are added as edges.
- **Minimizing Transfers**: The algorithm calculates the minimum number of transfers required to travel from the source station to other stations by processing the TPP graph and updating the transfer counts accordingly.

3 Implementation Summary

3.1 What We Implemented

We implemented three algorithms to minimize vehicle transfers:

- **SQ (Single Queue)**: The Single Queue (SQ) algorithm works by using a single queue to process the nodes in the graph. It starts with the source node and processes all its immediate neighbors, updating their minimum transfer count. After processing the neighbors, the algorithm revisits nodes and edges multiple times. Each time it revisits a node, it updates the minimum transfers if a better path (with fewer transfers) is found. This process is repeated until all reachable nodes are processed. While the SQ algorithm is simple, it can be slower compared to others, as it might revisit the same edges multiple times, leading to higher computational overhead, especially for larger datasets.
- **NQ (No Queue)**: The No Queue (NQ) algorithm is a more efficient approach than SQ because it does not use any queues. Instead, it processes each node level by level, ensuring that each node is visited exactly once. In this approach, the algorithm starts from the source node and updates the minimum transfer count for each of its neighboring nodes. The main advantage of NQ is that it avoids revisiting nodes and edges, resulting in a more efficient traversal. It processes the entire graph in a single pass, making it faster than SQ for larger datasets. However, this algorithm may still struggle with large and complex networks due to its dependency on the number of active edges and nodes.
- **MQ (Multiple Queues)**:The Multiple Queues (MQ) algorithm improves on both SQ and NQ by utilizing multiple queues, one for each level of the graph. This approach divides the graph into different levels based on the number of transfers

required to reach each node. Nodes that are at the same level are processed together in a queue, making it more efficient to process each level independently. The algorithm processes each node and its neighbors in parallel across different levels, reducing the need to revisit nodes repeatedly. MQ is particularly effective when the graph has distinct levels and can provide a better runtime for large graphs compared to SQ, though it can also be slower than NQ,and this was also one of the key findings of ours throughout this project.

3.2 Implementation Strategy

- The first step was to construct the temporal graph from the dataset.
- Next, we converted the temporal graph into a TPP graph.
- We then implemented the three algorithms and evaluated them on random datasets of varying sizes (small, medium, and large).

3.3 Difficulties Encountered

- **Data Generation**: We initially wanted to use the dataset that was used in the paper,and we invested a lot of time in it,but we were unable to either get the dataset that they used or any commute data of pakistan.Due to this,we had to settle for generating random datasets.And generating realistic temporal data with proper vehicle IDs and trip times was initially challenging. We had to ensure that the generated data followed realistic patterns for departure times and vehicle IDs.
- **Algorithm implementations**:it was also tough to implement all of these algorithms,we had to make sure that they were implemented right.It took a lot of time to debug them as well.

3.4 Changes from the Original Approach

In the original paper,there were 5 algorithms given,out of which we implemented the first 3 (other than the one that is used to convert a temporal graph into a Temporal paths perservation graph which we also implemented).They also used GTFS dataset,but since we were unable to get that,we used randomly generated datatsets of three sizes:large datasets had about 500 trips,medium dataset had about 50 trips and the small dataset had about 5 trips.

4 Evaluation

4.1 Correctness

The correctness of our implementation was validated through testing on small, medium, and large randomly generated datasets. The results were compared with theoretical expectations, and the minimum number of transfers was accurately calculated for each source-destination pair.

4.2 Runtime & Complexity

- For **SQ**: $O(n \cdot \Delta)$
- For **NQ**: $O(n + m)$
- For **MQ**: $O(r + L)$

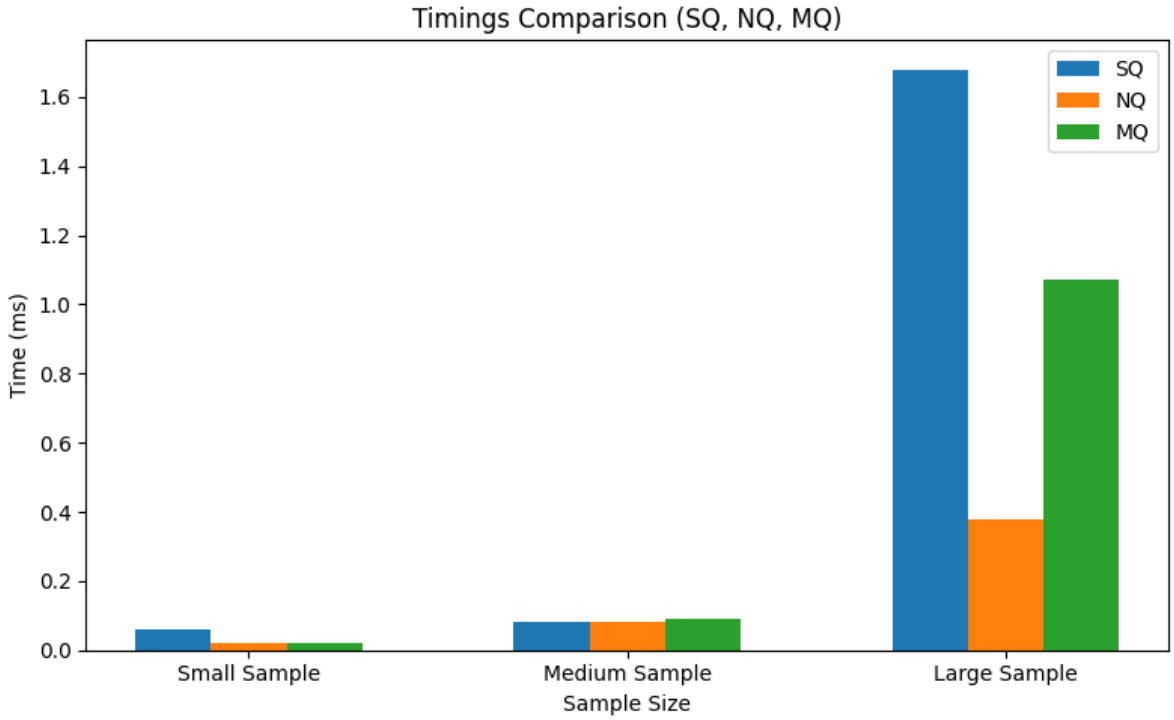
Where:

- n = number of stations (nodes)
- m = number of trips (edges)
- r = number of levels
- L = number of trips (edges)

4.3 Comparisons

There have been no algorithms that directly focus on reducing the numbers of transfers.Hence we compared the 3 algorithms that we implemented amongst themselves.

What we found was that on all of the three datasets,the NQ algorithm gave the best performance and took the least amount of runtime,as suggested by this graph below that plots the results we got:



5 Reflection

5.1 Learning Outcomes

- Through this project, we gained valuable insights into how graph theory can be applied to solve real-world problems, particularly in the context of public trans-

portation optimization. By modeling the transport system as a temporal graph, where each edge represents a trip with associated times and vehicle IDs, we were able to explore how graph algorithms can be used to minimize the number of vehicle transfers for passengers. This process involved translating theoretical concepts into practical, executable solutions, allowing us to develop a deeper understanding of algorithm design and optimization techniques.

- We also learned how to design and implement algorithms that account for time-dependent constraints, a key feature of transportation systems, such as departure times, travel durations, and vehicle changes. This made us more proficient in tackling problems that involve real-time data, which is a critical aspect of modern transportation systems.

5.2 Ideas for Future Work

- We can Implement Priority Queue (PQ) and Multiple Priority Queues (MPQ) to further optimize the algorithm for large-scale networks and then compare their performances.

- we can try Testing the algorithm on real-world datasets like GTFS data to validate its effectiveness in actual transit systems.

6 Conclusion

- This project successfully demonstrates the practical application of graph algorithms in optimizing public transportation systems, with a specific focus on minimizing vehicle transfers. By utilizing a temporal graph model, where each edge represents a trip between stations with associated departure times, durations, and vehicle IDs, we were able to design and implement an algorithm that efficiently calculates the minimum number of transfers needed to travel between stations. We implemented, tested, and validated three key algorithms—SQ (Single Queue), NQ (No Queue), and MQ (Multiple Queues)—each with its own approach to handling the graph traversal and transfer minimization.

- The algorithms were tested on randomly generated datasets of varying sizes (small, medium, and large), which allowed us to evaluate their performance in terms of runtime and correctness. The results showed that while SQ (Single Queue) was the simplest to implement, it had higher computational costs due to revisiting nodes multiple times. In contrast, NQ (No Queue) proved to be faster for larger datasets by processing nodes once, and MQ (Multiple Queues) offered an effective solution by processing nodes in levels, making it well-suited for networks with distinct transfer stages.

- We validated the correctness of the algorithms by comparing the calculated minimum transfers against theoretical expectations and verified that each algorithm correctly minimized transfers while adhering to temporal constraints.

- Looking forward, we plan to extend the project by applying the algorithms to real-world transportation data (like GTFS data), which would offer a more realistic test

of their effectiveness in actual transit systems. This real-world application will also allow us to fine-tune the algorithms, optimizing their performance for large-scale datasets, and potentially developing new algorithms that account for additional complexities such as traffic conditions, delays, or real-time route changes.

7 References

1 Sharma, A., & Kumar, R. (2024). Minimizing Transfers in Public Transportation Networks Using Temporal Graphs*. *Proceedings of the IC3 Conference, August 08–10, 2024, Noida, India.