Technical Summary Report:
Minimum Transfers in Public Transportation

Project Checkpoint 2,by Ghulam Mustafa and Syeda Wania

1 Problem and Contribution

The paper addresses the **Minimum Transfers Problem** in public transportation networks: determining the minimum number of vehicle changes (transfers) needed to reach all destinations from a source location. This problem is crucial for improving passenger experience, especially for vulnerable populations.

The main contribution is a novel data structure called the **Temporal Paths Preservation Graph (TPP-Graph)**, designed to maintain only time-respecting paths in temporal graphs. Along with this structure, the authors propose five algorithms to efficiently compute minimum transfers from one-to-all nodes.

2 Algorithmic Description

A public transportation network is modeled as a temporal graph $G = (V, E)$, where each edge represents a vehicle trip and contains a time and vehicle ID. The TPP-graph $\tilde{G}$ transforms each edge in G into a node and connects two nodes if their corresponding trips are time-respecting.

Here,we represent valid paths through edges, then assign weights: 0 for same-vehicle edges, 1 for different-vehicle edges. This ensures that any path weight in $\tilde{G}$ equals the number of transfers.

Algorithms

- **Single Queue (SQ):** This algorithm uses a single queue, similar to a line at a counter. It starts from all trips that begin at the source station and explores all connected trips. It keeps updating paths whenever it finds a smaller number of transfers. However, it may process the same trip multiple times, making it less efficient.
- **No Queue (NQ):** This method avoids using any queue. Instead, it processes trips level by level. A level represents all trips that can be reached with the same number of transfers. It activates trips in each level and updates their connections in the next level. Since each trip is checked only once, it's more efficient than SQ.
- **Multiple Queues (MQ):** This algorithm creates a separate queue for each level. It only processes trips that are actually reachable in that level, skipping unnecessary ones. This reduces repeated work and improves performance over the previous two methods.
- **Priority Queue (PQ):** Instead of going through levels one by one, this algorithm uses a priority queue to always process the most relevant level next. If a level has no active trips, it is skipped. This helps save time, especially in large networks with many levels.
- **Multiple Priority Queues (MPQ):** This is an optimized version that uses one priority queue per level. It keeps track of whether each trip has already been processed to avoid duplicates. This algorithm combines the benefits of MQ and PQ and reduces overhead from unnecessary checks.

Each algorithm begins with marking source edges, initializing transfer counts, and traversing $\tilde{G}$ to update the minimum transfer values.

3 Comparison with Existing Approaches

Previous studies focused on minimizing edge counts or travel time, but overlooked the real-world issue of vehicle transfers. Some optimized scheduling (design-phase), but not routing (operational-phase). Compared to shortest path or Dijkstra-like algorithms, this approach is:

- Better suited for transfer minimization.
- Operates on transformed graphs that eliminate invalid paths.
- Proven through correctness theorems and real-world datasets.

4 Data Structures and Techniques Used

- **Temporal Graphs:** Models transport networks with time and vehicle info.
- **TPP-Graph:** Ensures time-respecting paths only.
- **Priority Queues & Level Arrays:** Used in different algorithm variations for traversal efficiency.

5 Implementation Outlook

Challenges include:

- **Scalability:** Large datasets,like having millions of edges can stress memory and runtime.
- **Edge Validity Checks:** Ensuring edges meet time constraints adds computational overhead.
- **Algorithm Complexity:** PQ and MPQ approaches need some careful management of levels and priorities.