Optimizing Seattle Bus Transport Routes With Minimum Spanning Graphs

Giorgio Micaletto
Federico Giorgi
Taylor Woodward

Abstract

This project develops an optimized framework for redesigning Seattle's bus transit network using a graph-based approach informed by real-world Average Daily Traffic (ADT) data. The process begins by constructing a detailed road network graph, where edges are weighted according to road length, normalized busyness, convenience, and estimated transit frequency. We then generate a Minimum Spanning Tree (MST) from this weighted graph, providing a foundational structure that ensures connectivity with minimized operational costs. To enhance network efficiency and coverage, strategic redundant edges connecting major transit hubs are selectively reintegrated, resulting in an augmented network. Routes are initially determined based on hub scores and edge demands, after which an iterative route-merging algorithm reduces redundancy by consolidating geographically proximate routes while considering geometric and operational penalties. Lastly, our model systematically identifies optimal bus stop locations along these refined routes to maximize accessibility and convenience for riders.

Contents

Description of the problem **2**
Motivation . . . 2
Objectives . . . 2

Modeling the problem **2**
Data Acquisition and Overview . . . 2
Graph Modeling . . . 2
Data Processing and Integration . . . 3
Simplifications . . . 3
Graph Visualization and Analysis . . . 4

Solving the Problem **4**
Graph Construction and Edge Weight Assignment . . . 5
Route Merging and Geometric Optimization . . . 5
Transit Network Optimization and Route Count Reduction . . . 5
Bus Route Generation and Stop Placement . . . 7

Results and Visualization of the Network **7**

Limitations and Potential Improvements **8**

Appendix **10**

Description of the problem

Motivation

Urban centers worldwide are grappling with increasing congestion and its associated economic, environmental, and social costs. As populations grow and urban density intensifies, efficient public transit networks become crucial for sustainable city development. Despite the availability of extensive transportation data, many cities still face challenges in optimizing existing transit infrastructure to effectively meet user demand and maximize operational efficiency. The complexity of transit network design arises from numerous interacting factors, such as route coverage, frequency, and the strategic placement of stops to ensure maximum accessibility. Discrete mathematical models—particularly those rooted in graph theory and network optimization—offer powerful tools to address these challenges systematically [1, 2]. Designing an effective public transit network entails critical decisions on:

1. Selecting roads or corridors that will best serve rider demand.
2. Determining optimal frequencies and schedules for transit services.
3. Strategically placing stops to balance coverage and operational efficiency.

While the complexity of such tasks is considerable, structured, algorithmic approaches enable planners to develop networks that balance cost-efficiency with high-quality service.

Objectives

The primary objectives of this project are as follows:

1. **Develop a robust graph-based framework** to systematically analyze, visualize, and optimize urban bus transit networks.
2. **Identify critical urban areas with high transit demand**, ensuring frequent and reliable service to maximize network accessibility and efficiency.
3. **Construct a Minimum Spanning Tree (MST)** as a foundational network structure, integrating strategic redundant edges and route-merging techniques to **reduce redundancy and optimize transit coverage**.
4. **Systematically generate and position bus stops** to maximize rider convenience, accessibility, and operational efficiency.

By achieving these objectives, the model aims to provide transit planners with a robust and flexible framework, enabling informed decision-making and facilitating continuous improvements in urban transportation network design.

Modeling the problem

Data Acquisition and Overview

To construct the model of Seattle's transit network, we utilize publicly available traffic flow data from the **Seattle City GIS portal**. Specifically, we use the 2022 Traffic Flow Counts dataset [3], which provides a comprehensive record of vehicle volume counts across various streets and intersections in Seattle. The dataset includes key attributes:

- **Segment ID**: A unique identifier for each road segment.
- **Location Description**: The street name or intersection where the measurement was taken.
- **Direction of Travel**: Indicates whether the measurement is for northbound, southbound, etc.
- **Daily Traffic Volume**: The total number of vehicles recorded on a given segment per day.

Graph Modeling

We model the transportation network as a graph $G = (V, E)$, where:

- **Nodes (V)** represent intersection points or junctions, each assigned a **hub score** indicating its relative busyness, computed by aggregating the normalized busyness of connecting edges.
- **Edges (E)** represent road segments, characterized by attributes:
  - **Length (L_e)**: Physical length of the road segment.
  - **Normalized Busyness (b_e)**: Represents average daily traffic normalized between 0 and 1.

To optimize the transit network, we define an **edge weight function** $w : E \to \mathbb{R}^+$ as follows:

$$w(e) = L_e \left(1 + \lambda_m |b_e - 1| + P_c\right) - \beta(h_s + h_t)\frac{\bar{L}}{10} + \gamma f_e$$

where:

- L_e is the length of edge e.
- b_e is the normalized busyness of edge e.
- λ_m penalizes deviations from optimal busyness.
- P_c is a convenience penalty:

$$P_c = \begin{cases} \lambda_c \cdot \frac{L_e}{\bar{L}}, & \text{if } h_s > \tau \text{ and } h_t > \tau \\ 0, & \text{otherwise} \end{cases}$$

  with:
  - λ_c: Convenience penalty factor.
  - $\bar{L}$: Average road length in the network.
  - τ the threshold defining a busy hub.
- h_s and h_t are hub scores at the edge endpoints.
- β is the hub bonus factor, rewarding edges connecting busy hubs.
- f_e is the estimated frequency of transit services, computed as:

$$f_e = \text{base\_freq} + S \cdot \frac{(b_e + h_s + h_t)}{3}$$

- γ is a fare-related cost coefficient, scaling with estimated frequency.

This formulation captures key factors—distance, busyness, hub connectivity, service frequency, and fare-related costs—to achieve optimal balance in transit network design.

Data Processing and Integration

To integrate the provided geospatial traffic data into our previously established graph-based optimization framework, we perform comprehensive preprocessing, outlined as follows:

1. **Geospatial Mapping**: We use Geographic Information System (GIS) shapefiles, specifically a GeoJSON dataset containing detailed spatial information about road geometries, intersections, and existing network characteristics. Each road segment is extracted and mapped as an edge within our graph-based representation of the transportation network, enabling precise alignment between spatial coordinates and graph nodes and edges.
2. **Coordinate System Standardization**: Accurate spatial analysis requires consistent units of measurement. Hence, we project geographic coordinates into an appropriate projected coordinate system (EPSG:26910), ensuring accurate calculations of road lengths and spatial distances critical to subsequent analyses and optimizations.
3. **Aggregation and Normalization of Traffic Data**: We utilize the Average Daily Traffic (ADT) data included in the dataset to quantify the relative busyness of road segments. These values are then normalized to a standardized busyness score $b_e \in [0, 1]$, providing an intuitive metric for assessing traffic load. A higher b_e indicates greater traffic volume, thereby guiding network optimization towards prioritizing service coverage and frequency in high-demand areas.

Simplifications

Real-world transit modeling inherently involves complexities that are challenging to precisely address, particularly due to limitations in data quality, computational resources, and the inherent variability of urban environments. To maintain computational tractability and efficiency, we have strategically implemented several simplifications in our modeling framework:

- **Temporal Traffic Variability**: Our analysis exclusively utilizes Average Daily Traffic (ADT), the only available dataset. Consequently, hourly (peak versus off-peak) and weekly (weekday versus weekend or holiday) variations in traffic flow are omitted, potentially reducing the granularity and temporal accuracy of route optimization results.
- **Uniform Vehicle Types**: Our framework does not differentiate between vehicle categories such as private vehicles, commercial trucks, or existing public transit vehicles. All vehicles contribute equally to the normalized busyness scores, which simplifies calculations but does not fully capture the operational realities where commercial or freight traffic may have distinct routing and infrastructural needs.
- **Static Network Conditions**: We model the network as static and unchanging, omitting temporary dynamics such as road closures, construction activities, or accident-induced disruptions. Although simplifying, this assumption limits the model's adaptability to real-time operational adjustments.
- **Uniform Service Frequencies**: Service frequencies are estimated using average busyness and hub scores without explicitly integrating demographic, socioeconomic, or ridership-specific data. This simplification facilitates a generalizable and computation-

ally efficient approach, though it may overlook localized demand variations driven by community-specific characteristics.

- **Homogeneous Stop Placement**: Bus stop placement considers only spatial proximity, merging nearby stops based on distance thresholds without accounting for subtler operational considerations such as passenger safety, sidewalk accessibility, curbside management constraints, or existing infrastructure features (e.g., pedestrian crossings, traffic signals).
- **Route Geometry and Comfort Factors**: Our geometric penalties primarily account for route continuity and turning angles. However, the model does not incorporate rider comfort factors explicitly, such as smoothness of rides, acceleration patterns, or specific roadway conditions, all of which influence passenger experience and overall transit attractiveness.
- **Absence of Capacity Constraints**: The model assumes unlimited bus capacity and does not explicitly handle scenarios involving overcrowding or variable passenger load distributions, factors critically important for practical scheduling and frequency planning.

These simplifications facilitate an analytically manageable and computationally practical framework, enabling us to clearly highlight essential structural and operational aspects of transit networks. Despite these simplifications, our model provides valuable insights for strategic network optimization, serving as a foundation that could be enriched further with additional data and more detailed modeling approaches.

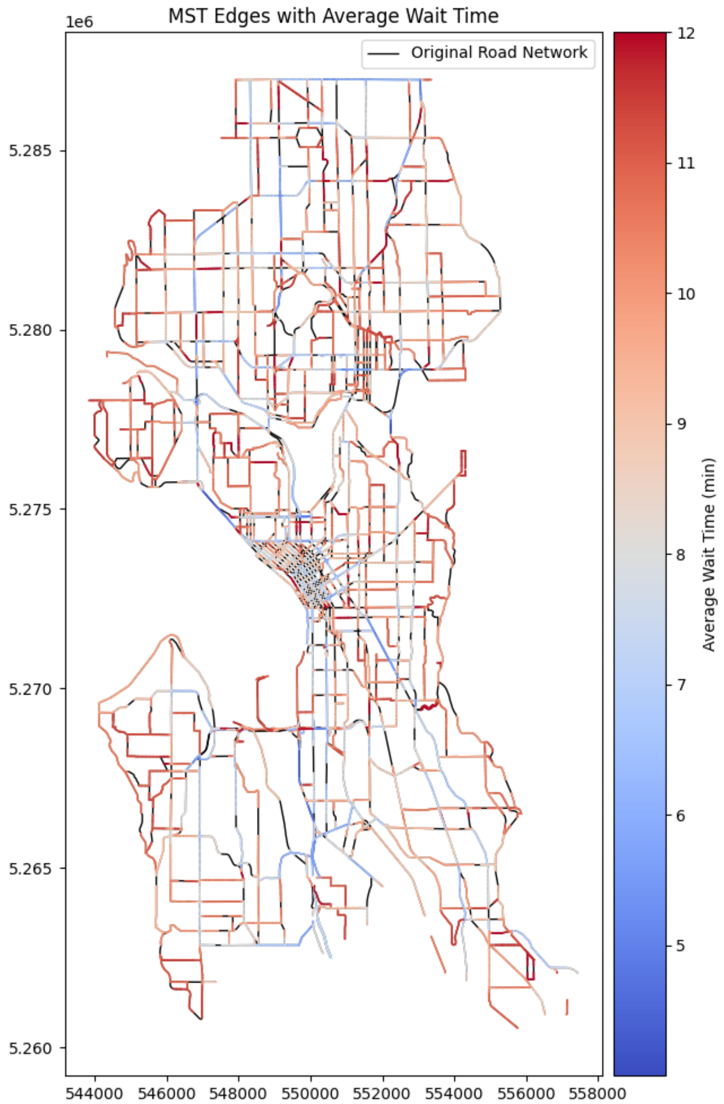

Figure 1: *Heatmap of average waiting times on road segments selected by the MST algorithm, highlighting transit efficiency across the network.*

Graph Visualization and Analysis

Figure 1 illustrates a visual representation of the road network generated through our Minimum Spanning Tree (MST) algorithm. Each road segment within the MST is colored according to its estimated average wait time for transit service, employing a gradient from blue (shorter wait times) to red (longer wait times). This visualization enables a clear and immediate understanding of service efficiency distribution across the city's road network. By highlighting segments that might experience higher congestion or lower service frequency, planners can quickly identify priority areas requiring enhanced transit solutions. Moreover, the MST-based representation serves as an efficient foundational structure, ensuring comprehensive city-wide connectivity while minimizing the total operational cost.

Solving the Problem

To systematically approach bus route optimization, we developed a structured, multi-stage framework grounded in graph theory and network optimization methodologies.
The computational framework leverages the following Python libraries:

- **NetworkX**: Graph representation, analysis,

and MST computations.
- **GeoPandas**: Processing and spatial analysis of geospatial road data.
- **Matplotlib** and **Plotly**: Visualization of routes and service metrics.
- **NumPy** and **math**: Numerical calculations, geometric computations, and normalization processes.

Graph Construction and Edge Weight Assignment

We model Seattle's transportation network as a detailed, weighted graph $G = (V, E)$, where each node $v \in V$ corresponds to road intersections or junctions, and each edge $e \in E$ represents distinct road segments connecting these intersections. Edges in this graph carry attributes essential for transit optimization, such as length, normalized busyness derived from Average Daily Traffic (ADT), and geometry data.

The graph is constructed using Algorithm 1, which systematically assigns weights to edges based on several critical factors:

- **Road Length**: Represents the physical length of each segment, influencing travel times and operational costs.
- **Normalized Busyness**: Derived from ADT values, indicating traffic demand and influencing transit frequency decisions.
- **Hub Connectivity**: Each node is assigned a *hub score*, reflecting its importance in terms of network connectivity and traffic demand. This score aggregates the normalized busyness of all incident edges.
- **Service Frequency Estimation**: Derived from normalized busyness and hub scores, guiding decisions on transit service intensity.

By integrating these attributes, our graph model effectively captures the complexities of urban transit demands, providing a solid foundation for further optimization.

Route Merging and Geometric Optimization

To enhance the efficiency of transit routes and minimize redundancy, we developed an advanced route merging algorithm. This algorithm systematically evaluates and merges pairs of transit routes based on proximity, geometric continuity,

Algorithm 1 Graph Construction and Edge Weight Computation

Initialize an empty graph G
for all road segments r in road dataset **do**
  Extract spatial geometry, traffic busyness (ADT), and segment length
  Normalize busyness scores for consistency across segments
  Create edges in graph G with these attributes
end for
for all nodes v in graph G **do**
  Compute hub scores as aggregated normalized busyness from incident edges
end for
for all edges (u, v) in graph G **do**
  Calculate edge weights incorporating length, normalized busyness, endpoint hub scores, convenience penalties, and estimated service frequency
end for
Generate Minimum Spanning Tree (MST) from weighted graph G
return MST, hub scores, and the full graph G

overall route length, and network-specific constraints, including penalties for undesirable route characteristics (such as excessive turning angles and use of restricted edges). The approach prioritizes merging routes that will produce the lowest combined geometric and operational cost, ensuring optimal route continuity and practicality.

Algorithm 2 details this merging process, and by systematically applying it, our model effectively reduces redundant overlaps, optimizes route geometry, and ensures practical and user-friendly transit routes.

Transit Network Optimization and Route Count Reduction

To enhance the practical effectiveness of our transit network, we expand the foundational Minimum Spanning Tree (MST) by reintroducing selected redundant edges that improve connectivity and service reliability at high-demand locations. This augmented network better represents real-world transit requirements, accommodating critical routes and reducing vulnerability to service disruptions. Subsequently, we employ an iterative algorithm to systematically reduce the overall number of bus routes without compromising ser-

Algorithm 2 Route Merging and Optimization Algorithm

Initialize variables: best_cost $\leftarrow \infty$, best_merge $\leftarrow$ None
for all pairs of candidate routes $(\text{route}_1, \text{route}_2)$ **do**
  Identify the closest pair of endpoints between route_1 and route_2
  Compute the shortest feasible connecting path considering network penalties, including forbidden or restricted edges
  Calculate the total cost of the merged route, incorporating factors such as merge segment length, geometric continuity (e.g., turning angle penalties), and penalties for restricted edges
  if calculated merge cost is lower than the current best cost **then**
    Update best_cost and best_merge with the newly identified optimal merge
  end if
end for
return optimal merged route (best_merge) and associated merge cost (best_cost)

vice quality or accessibility.

Algorithm 3 elaborates on the transit optimization and route reduction process:

Algorithm 3 Transit Optimization and Route Count Reduction

Input: MST graph, original full road network graph G, hub scores
Output: Optimized and reduced set of transit routes
Initialize augmented_network as a copy of MST
for all edges (u, v) in MST **do**
  Estimate transit frequency based on normalized busyness and hub scores at endpoints
  Assign average wait time attribute as $60/f_e$, derived from estimated frequency
end for
Define a redundancy threshold for selecting critical redundant edges based on node hub scores
for all edges (u, v) in original graph G not present in MST **do**
  if hub scores for nodes u and v both exceed redundancy threshold **then**
    Calculate estimated transit frequency and assign corresponding wait time attributes
    Integrate edge into the augmented transit network
  end if
end for
Generate an initial set of bus routes based on calculated transit frequencies and network demand
while the number of routes exceeds the desired limit **do**
  Identify candidate route pairs for merging based on geographic proximity and connectivity
  Calculate merge costs by considering factors such as merged route length, geometric continuity penalties, and operational constraints
  Select the candidate pair with the lowest total merge cost
  Merge selected routes and update the route list accordingly
end while
return Final optimized set of transit routes

Through this combined approach, we ensure an efficient, practical, and resilient transit network design capable of effectively responding to urban transportation needs.

Bus Route Generation and Stop Placement

In the final step, we explicitly generate bus routes by leveraging computed edge demand, ensuring routes closely align with anticipated passenger needs. Strategic bus stop locations are determined systematically to enhance rider accessibility and convenience. Algorithm 4 details this integrated approach, combining route creation and optimized stop placement:

Algorithm 4 Bus Route and Stop Placement Generation

1: **Input**: Augmented transit network graph with edge demand attributes
2: **Initialize**: empty list of routes
3: **while** there are edges with remaining transit demand **do**
4: Identify the edge with the highest remaining demand as starting route segment
5: Iteratively extend routes from endpoints by adding adjacent edges with highest remaining demand
6: Append completed route to the final route list
7: **end while**
8: Identify mandatory stop locations: intersections, route endpoints, and high-demand junctions
9: Insert additional stops along each route segment if the distance between mandatory stops exceeds predefined spacing thresholds
10: Merge nearby stops to consolidate stop locations, applying spatial tolerance criteria
11: **Output**: Finalized bus routes and global set of optimized stop locations

This explicit process ensures that the resulting bus network is not only structurally efficient but also rider-centric, maximizing accessibility and convenience across the transit network.

Results and Visualization of the Network

Our methodology begins with the construction of an initial Minimum Spanning Tree (MST) derived from Seattle's extensive road network, creating a foundational structure that identifies essential urban connections with minimal redundancy. Figure 1 illustrates this MST, presenting a heatmap where segments are color-coded according to their estimated average transit wait times. Cooler colors (blue) represent shorter wait times, while warmer colors (red) indicate areas with higher anticipated waiting periods. This visualization enables clear identification of critical congestion points and supports informed decision-making for further route optimization.

Subsequent iterative steps, including sophisticated geometric route merging algorithms, are applied to enhance this foundational MST structure. Through these optimization iterations, we systematically reduced the initially generated routes—originally numbering over 1,000—to a final targeted set of approximately 300 highly optimized routes. This reduction significantly improves operational manageability while maintaining comprehensive transit coverage throughout Seattle.

A primary operational goal of our model is to achieve an average waiting time of approximately 8 minutes per passenger, ensuring a balance between operational efficiency and rider convenience. Figure **??** illustrates an optimized example route resulting from this approach, showcasing how carefully calculated merging and refinement produce practical, contiguous, and accessible transit paths. Figure 3 offers a detailed view of strategically placed bus stops along one of these optimized routes, highlighting precise decisions in stop placement that maximize accessibility and rider convenience.

Though our optimized network is theoretical, it provides valuable insights and a structured foundation for transit planners. It identifies key areas for improved frequency and connectivity, and by carefully balancing the complexities of transit planning, our model significantly enhances potential service reliability and accessibility. Moreover, the modular design of our framework ensures adaptability, allowing easy adjustments based on evolving urban conditions or new data availability.

In conclusion, our approach successfully integrates rigorous network optimization techniques with practical considerations, providing an effective, scalable, and actionable strategy for improving urban public transportation networks.

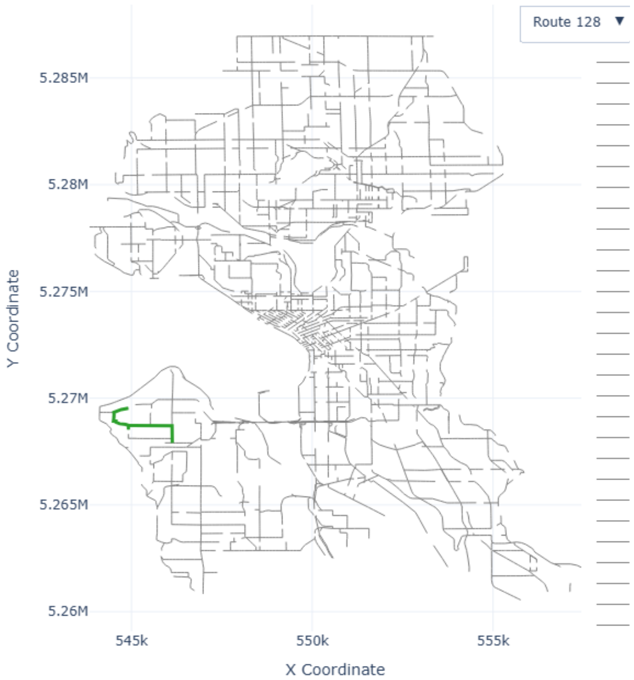

Figure 2: *View of one of the optimized routes.*

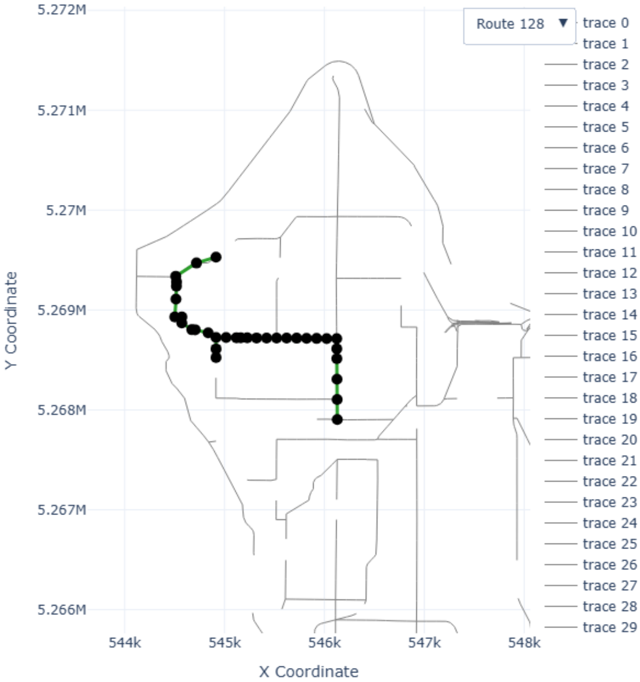

Figure 3: *Detailed view of the optimized route, highlighting strategically positioned bus stops to maximize rider accessibility and service efficiency.*

Limitations and Potential Improvements

While our proposed transit optimization framework effectively balances complexity and practical implementation, it inherently includes simplifying assumptions and limitations that restrict its direct applicability in real-world operational settings. Identifying and addressing these limitations presents clear pathways for future improvements:

Route Realism and Geometric Continuity Currently, our model occasionally generates transit routes exhibiting unconventional geometric shapes—such as tree-like branching—that differ significantly from typical linear or loop-based transit lines commonly found in practice. These non-standard geometries can introduce practical challenges, including increased operational complexity, difficult bus navigation, and decreased rider intuitiveness. Enhancing route realism through algorithms that favor simpler, more conventional geometric structures and minimize complex maneuvering (such as frequent U-turns) would significantly improve the operational feasibility and rider experience.

Computational Efficiency and Scalability The existing implementation, particularly in route merging and stop placement, involves extensive iterative computations and nested spatial analyses, leading to computational inefficiencies when applied to larger networks. Future work should explore optimized data structures (e.g., spatial indexing, efficient caching), vectorized numerical operations using libraries such as NumPy, or parallelized computation frameworks. Leveraging parallel processing or distributed computing approaches would greatly enhance the model's scalability and responsiveness to larger datasets, essential for real-time applications and practical deployments.

Incorporation of Dynamic Traffic Conditions Our analysis presently relies solely on static Average Daily Traffic (ADT) data, excluding temporal dynamics like rush-hour peaks, weekend traffic variations, and seasonal fluctuations. Integrating temporally segmented traffic data, potentially through real-time data streams or predictive traffic modeling, could enable adaptive route and frequency adjustments, making transit services more responsive and accurately tailored to fluctuating passenger demands throughout the day and week.

Integration of Detailed Demographic and Socioeconomic Insights Service frequency and route coverage are currently determined primarily by normalized busyness scores and hub scores, neglecting detailed demographic and socioeconomic factors that critically influence transit demand. Enhancing the model by explicitly incorporating population density, commuter patterns, socioeconomic status, and employment hubs could refine transit service frequencies and route prioritization, enabling highly targeted, equitable, and effective service provision.

Vehicle Capacity Constraints and Comfort Considerations At present, the model does not explicitly account for vehicle capacity limits or comfort considerations such as ride smoothness, acceleration/deceleration patterns, and passenger load distributions. Addressing these factors explicitly could improve practical scheduling and frequency allocations, reducing overcrowding and improving passenger comfort, thus increasing overall transit attractiveness.

Advanced Stop Placement Considerations The current approach to stop placement focuses predominantly on spatial proximity and convenience, merging stops based on distance alone. Real-world operational constraints, such as pedestrian safety, sidewalk accessibility, curbside management, existing infrastructure, and interaction with traffic signals, significantly impact optimal stop positioning. Accounting for these nuanced factors would improve stop usability, passenger safety, and overall system effectiveness.

Comparative Analysis and Validation Finally, our theoretical model lacks direct validation against an existing operational baseline, limiting empirical verification of its effectiveness. Conducting comparative analyses with current transit networks or simulated environments would provide clearer insights into actual performance gains and practical limitations, informing more effective and realistic future deployments.

Incorporating these improvements would elevate the current framework from theoretical robustness towards practical operational effectiveness, offering comprehensive, realistic, and actionable transit solutions tailored to evolving urban transportation needs.

References

[1] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[2] T. H. Cormen et al. *Introduction to Algorithms*. 3rd. MIT Press, 2009.

[3] Seattle Department of Transportation. *2022 Traffic Flow Counts Dataset*. 2022. URL: `https://data-seattlecitygis.opendata.arcgis.com/datasets/SeattleCityGIS::2022-traffic-flow-counts/about`.

Appendix

Coordinate Precision and Vertex Identification Due to floating-point precision limitations, we implement a rounding strategy when creating vertex identifiers. Specifically, vertex coordinates are rounded to a defined decimal precision (e.g., three decimals) to ensure consistent vertex matching across road segments and intersections. This approach mitigates errors arising from numerical inaccuracies.

Geometric Penalties and Route Shape Optimization Our geometric penalty function penalizes routes for two primary reasons:

- **Route Openness**: Routes that start and end far apart are penalized, promoting closed-loop structures or effectively looped routes to minimize operational complications.
- **Turning Angle Penalties**: Excessive turning angles at intersections contribute to increased operational complexity and reduced rider comfort. The penalty is computed by averaging absolute angles across route segments.

Normalization of Traffic Busyness Normalized busyness scores are computed by linearly scaling the Average Daily Traffic (ADT) values between 0 and 1. The normalization formula is:

$$b_e = \frac{ADT_e - \min(ADT)}{\max(ADT) - \min(ADT)}$$

This normalization facilitates standardized comparisons and consistent frequency estimation across different road segments.

Estimation of Transit Frequency Transit frequency estimation integrates multiple data attributes (busyness, hub scores) according to the formula:

$$f_e = \text{base\_frequency} + S \cdot \frac{b_e + h_s + h_t}{3}$$

where S is a scaling factor, and base frequency ensures a minimal acceptable service frequency. The purpose is to systematically balance transit availability with expected passenger demand.

Geospatial Coordinate System Justification The chosen projected Coordinate Reference System (CRS) EPSG:26910 (UTM Zone 10N) provides accurate distance measurements essential for precise geometric and route calculations. Geospatial data originally provided in geographic coordinates (latitude and longitude) must be transformed to this CRS to preserve spatial accuracy during computations.

Route Merging Criteria Routes are merged based on a multi-factor cost function, emphasizing:

- **Spatial Proximity**: Routes are considered for merging if endpoints are within a specified distance threshold.
- **Operational Constraints**: Penalties for forbidden edges ensure route viability and operational compliance.
- **Geometric and Continuity Considerations**: A geometric cost metric evaluates the continuity, length, and directional coherence of merged routes.

Global Stop Placement Criteria and Merging Strategy Stops are initially placed at mandatory locations, including intersections, endpoints, and high-demand junctions. Additional intermediate stops are inserted at set intervals to maintain reasonable walking distances for riders. Spatial merging of nearby stops reduces redundancy and ensures practical stop distributions, maintaining operational efficiency and accessibility.

Software Implementation Details Our implementation utilizes Python and a combination of specialized libraries:

- **GeoPandas**: For spatial data handling and geospatial computations.
- **NetworkX**: To manage graph-based structures and perform network optimizations.
- **Matplotlib and Plotly**: For both static and interactive visualizations, enabling intuitive interpretation of results.
- **NumPy**: To efficiently handle numerical operations and vectorized calculations.

Future developments could extend the model with advanced spatial indexing and parallelization strategies to further enhance computational performance and scalability.
This appendix provides supplementary details crucial for comprehensive understanding, reproducibility, and future model improvements.