Optimization of the LPP Network

Ariana Kržan[a], **Nikolay Vasilev**[a], **Žan Pižmoht**[a], **and Timotej Zgonik**[a]

[a]University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia

The manuscript was compiled on May 19, 2025

Network optimization is essential to improve efficiency in various domains. It is especially important in public transport design, where we want to reduce road congestion, travel time, air pollution, and energy consumption. Enhancing public transport networks can significantly boost urban mobility, increase user satisfaction, and support a sustainable urban environment.

In this research, we try to optimize the Ljubljana Public Transport (LPP) network, which currently faces several inefficiencies, including congestion, suboptimal routes, and insufficient service coverage. By addressing these issues, we aim to develop a comprehensive and effective solution.

Our approach involves employing various optimization methods and techniques to redesign the LPP network. We aim to minimize travel time, reduce the number of transfers required between buses, and maximize service coverage.

Public transport network optimization plays a crucial role in enhancing urban mobility, reducing congestion, and improving overall efficiency. In this study, we will focus on optimizing the Ljubljana Public Transport (LPP) network and address its current inefficiencies.

Network optimization is a complex problem and is cruicial in various fields, inluding transportation. The primary function of these networks is effective routing of transport that can involve goods, information or passengers (1).

In the case of public transport, the networks can be multi-level, covering areas of both urban and interurban levels (2). However, in our problem of optimizing the LPP network, we will work with a single-level network, focusing on urban transportation within the city.

In network optimization, various techniques that solve problems such as the shortest path, max-flow, transportation, transshipment and vehicle routing have been developed over the years, incorporating both continuous and discrete models. (3) In the context of public transport, advanced heuristic methods, such as genetic algorithms, have proven effective in balancing user satisfaction with operational efficiency (4, 5).

The Ljubljana Public Transport network currently faces many challenges, including congestion, long traveling times, and inefficient routing. In recent years, despite improvements such as revamping the bus fleet, enhancing bus stops, and introducing separate bus lanes, the number of passengers has been declining. The main reasons for this are unfortunately not entirely clear. However, it is clear that travel times, a key quality indicator, have not been adequately studied or optimized (6).

For that reason, we will attempt to apply different optimization techniques to optimize the current LPP network and make it more efficient.

Related work

Optimization of public transport networks has been the object of several different studies, focusing on various methods to lessen congestion, reduce travel time and improve efficiency. In this section, we will review some related works.

Danila et al. (1) debate the traditional approach of shortest-path routing and how it is effective in reducing travel time. It, however, faces other issues such as congestion in highly connected nodes. Therefore, they suggest a heuristic transport routing optimiziation algorithm instead, which distributes traffic across the network more evenly and reduces bottlenecks while reducing travel time.

Bertsekas (3) reviews different optimization techniques for both continuous and discrete models. The book covers problems such as the shortest path, the max-flow, the min-cost flow and more, while describing different methods to solve them.

Next, Zhao and Ubaka (7) introduce a mathematical methodology for transit network optimization, which aims to reduce the number of transfers and optimize route directness while maximizing service coverage. They tested their methodology on a large-scale network in Miami-Dade County in Florida and discovered potential in systematic route planning.

Networks can be single or multi-leveled and Van Nes (2) explored the dependencies between urban and interurban transport networks.

Huang et al. (4) display a Geographical Information Systems based framework for bus network optimization using a genetic algorithm. This algorithm was also used by Bielli et al. (5), where they iteratively generated new bus network sets and evaluated them using performance indicators.

Yang et al. (8) designed a bus network optimization model based on the coarse-grain parallel ant colony algorithm. Fan and Machemehl (9) tried solving the problem by using a simulated annealing algorithm. This approach exceeded the use of genetic algorithms.

Serin and Mete (10) proposed a new model based on graphs for public transport networks, which aims to solve issues of a more traditional graph approach.

And lastly, the study of the Dutch Integral Transportation Study (11) describes a method that breaks the given problem into smaller sub-problems, which makes the original problem more computationally feasible for large networks.

Methodology

In graph theory, the **shortest path problem** describes a problem where we aim to find the minimum path length in a directed graph $G = (N, A)$ with nodes labeled as $1, \ldots, N$. Each directed arc $(i, j) \in A$ has an associated cost or length a_{ij}. The goal is to determine the shortest path from a starting node

to a destination node, which we accomplish by minimizing the sum of the costs along the path:

$$\sum_{n=1}^{k-1} a_{i_n i_{n+1}}, \qquad [1]$$

where k is the number of nodes in the path and the indices i_n and i_{n+1} represent consecutive nodes (3).

Our optimization approach uses **Dijkstra's algorithm** (1, 3) which is designed to solve the shortest path problem. It is the special case of the generic algorithm where the node `i` removed from the candidate list `V` at each iteration has minimum label, that is,

$$d_i = \min_{j \in V} d_j. \qquad [2]$$

To better understand the method, we will explain it explicitly. Initially, we have

$$V = \{1\}, \quad d_1 = 0, \quad d_i = \infty \quad \text{for all } i \neq 1. \qquad [3]$$

The method proceeds in iterations and terminates when V is empty. The typical iteration (assuming V is nonempty) is as follows:

Algorithm 1 Label Setting Method for Dijkstra's Algorithm.

Require: Graph G, source node s
Ensure: Shortest path distances d from source s

1: Initialize V with all nodes in G, $d_i = \infty$ for all $i \neq s$, $d_s = 0$
2: **while** V is not empty **do**
3: $i \leftarrow$ node in V with minimum d_i
4: Remove i from V
5: **for** each outgoing arc $(i, j) \in A$ **do**
6: **if** $d_j > d_i + a_{ij}$ **then**
7: $d_j \leftarrow d_i + a_{ij}$
8: **if** j is not in V **then**
9: Add j to V

We will draw heavily on the insights and concepts presented in the article by Serin and Süleyman ((10)). They present a novel public transportation graph named **ostensive public transportation graph (OPTG)**, which solves issues of existing public transportation network representations.

It does that by incorporating information of vehicle, time, fees and distance simultaneously. This information can be represented for transactional or analytical systems. Similarly we will consider not only the distance between bus stations, but also the different routes, buses, and timetables. Considering that the bus fare is 1.30 Euro for all buses in Ljubljana within Zone 1, there is no need to use the fee in the optimization.

Data Preprocessing

Out starting network consisted of stop names and connections between these stops. To refine the dataset for our analysis, we performed the following preprocessing steps.

We decided to focus our study predominantly on Zone 1 and removed most of the stops and connections pertaining to Zones 2 and 3 (see Figure 1), as well as those in rural communities in the eastern part of the municipality.

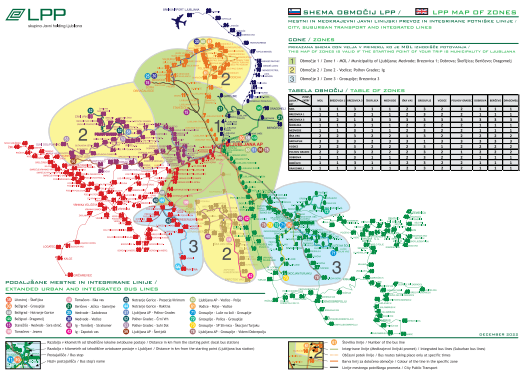

Fig. 1. LPP map of zones.

Next we added geolocation data for each stop using `google distance matrix api` (12), providing precise coordinates (Figure 2), and calculated the distances between each pair of connected stops in meters (Figure 3).

```
if mode == 'vertices':
    parts = line.strip().split("\"")
    vertex_id = int(parts[0].strip())
    station_name = parts[1].replace("\"", "")
    location = geolocator.geocode(f"{station_name}, Ljubljana, Slovenia")
    if location:
        lat, long = (location.latitude, location.longitude)
    else:
        no_slash_station_name = station_name.split("/")[0]
        location = geolocator.geocode(f"{no_slash_station_name}, Ljubljana, Slovenia")
        if location:
            lat, long = (location.latitude, location.longitude)
        else:
            location = geolocator.geocode(f"{station_name}, Slovenia")
            if location:
                lat, long = (location.latitude, location.longitude)
            else:
                missed_locations.append(station_name)
                lat, long = None, None
                print(f"Location not found found for {station_name}")

    G.add_node(vertex_id, name=station_name, lat=lat, long=long)
```

Fig. 2. Code snippet for assigning geolocation.

```
from geopy.distance import geodesic
from tqdm import tqdm

def calculate_distance(coord1, coord2):
    return round(geodesic(coord1, coord2).meters)

i = 0
for u, v, data in G.edges(data=True):
    if 'lat' in G.nodes[u] and 'lat' in G.nodes[v]:
        coord1 = (G.nodes[u]['lat'], G.nodes[u]['long'])
        coord2 = (G.nodes[v]['lat'], G.nodes[v]['long'])
        distance = calculate_distance(coord1, coord2)
        G[u][v][0]['distance'] = distance
    else:
        print(f"Geo data not found for one of the nodes: {u}, {v}")

print(i)
```

Fig. 3. Code snippet for distance calculation.

Alongside our original data, we also extracted additional data from the Web about different buses in LPP, as well as their timetables and routes. This additional data allowed us, as we already mentioned, to explore more complex optimization techniques by integrating multiple attributes, such as distance, time, bus types, and routes.

Implementation

After preprocessing our data, we end up with a multigraph (10), where each pair of nodes represents different connected stations and may have multiple connections between them, one for each bus that travels between the two selected stops.

This graph we put in our optimization algorithm, which is visible in Figure 4. The algorithm first transforms the undirected multi-graph G into a weighted multigraph $weighted\_G$ by integrating the external transit data. Nodes in G are mapped to corresponding transit stops, and datasets detailing stops, trips, routes, and timings. The method finds the soonest available bus trips to determine connectivity, applying a constraint to limit computational load. Edge weights are calculated as a linear combination of physical distance and travel time, with the weighting factor α balancing these dimensions. The resulting graph $weighted\_G$ serves to identify the k-shortest paths between specified nodes, optimizing routes based on both distance and expected travel duration. This is why it is an input for the shortest paths function in Figure 5.

Fig. 4. Code snippet of optimization function.

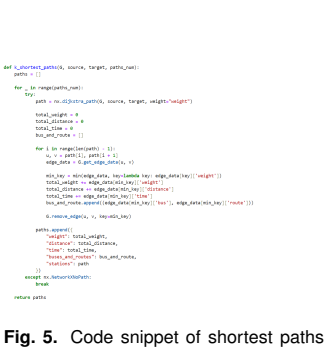

Fig. 5. Code snippet of shortest paths function.

This function iteratively calls the networkx's Dijkstra algorithm in order to extract the shortest path based on the precalculated weight. Then the extracted shortest path is removed in order to allow the function to find the next shortest path. This process is repeated until the specified number of shortest paths is reached or there are no more paths available from the source to the target node in the weighted graph.

Results

To demonstrate the results of the function, we will use as an example a trip from station Nove Jarše-Šmartinska to Leskoškova. Using different values of $\alpha = \{0, 0.25, 0.5, 0.75, 1\}$, we generated a weighted graph for the mentioned trip for each value of α.

The results of our optimization algorithm can be seen in Figure 6. On the left-hand side, we can see the first shortest path for each value of α based on the weight. For comparison, on the right-hand side is the second shortest path for each value of α. In this case, our weight is a combination of only 2 attributes - distance and time, and since the distance is pretty much the same for different buses, the time makes the bigger difference. For each of the extracted paths, we can see that different buses on different routes are suggested. As most stations are well-connected or only have a single line running through them, the generated paths have few differences. As there are only 2 attributes, it is expected that as α increases, time becomes the more important optimization parameter. With a combination of 3 or more attributes, we would expect the generated paths to display larger differences.

Discussion

In this project, we address the challenges of optimizing the LPP transport network and propose an algorithm that suggests new routes between selected bus stops based on weighted connections between the nodes. Our goal was to reduce travel time and minimize the number of transfers.

We began by collecting data and preprocessing it, which involved removing stops from zones 2 and 3, and adding geolocating data for each stop, enabling us to calculate the distances between them.

We continued by constructing a weighted multigraph and implementing an optimization algorithm. Our approach is inspired by Dijkstra's algorithm to solve the problem of shortest paths and generates ostensive public transportation graphs to model travel factors such as time, cost, and distance.

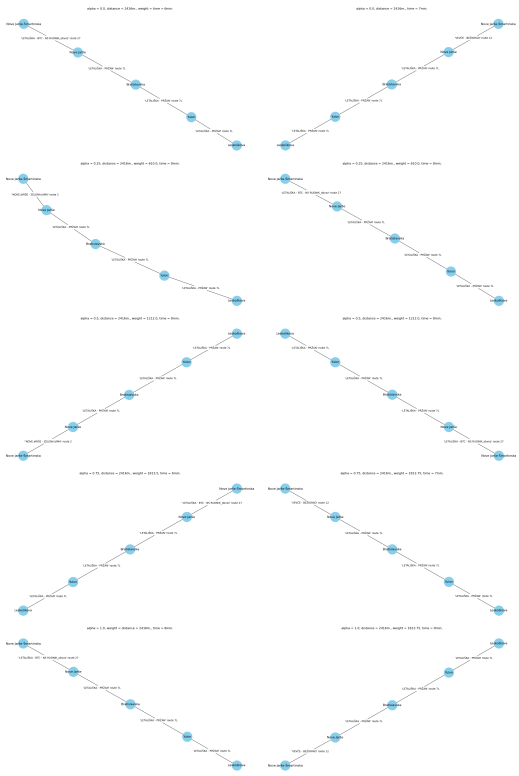

Fig. 6. Results for running the optimization algorithm from station Nove Jarše-Šmartinska to Leskoškova.

The results demonstrate that our optimization algorithm effectively adjusted routes in the LPP network by varying weights of distance and time through different values of α. This adjustment shows potential for enhancing route efficiency under diverse operational goals, such as minimizing travel times.

The algorithm's flexibility makes it a valuable tool for urban planners and transport authorities aiming to improve public transport services.

For future work, we suggest incorporating additional data such as traffic congestion, actual travel time, number of passengers, weather conditions, and more, which were not included in this study. That would also allow for implementation of more advanced algorithms and simulations.

1. Bogdan Danila, Yong Yu, John A Marsh, and Kevin E Bassler. Transport optimization on complex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 17(2), 2007.
2. Rob van Nes. Multilevel network optimization for public transport networks. *Transportation research record*, 1799(1):50–57, 2002.
3. Dimitri Bertsekas. *Network optimization: continuous and discrete models*, volume 8. Athena Scientific, 1998.
4. ZD Huang, XJ Liu, CC Huang, and JW Shen. A gis-based framework for bus network optimization using genetic algorithm. *Annals of GIS*, 16(3):185–194, 2010.
5. Maurizio Bielli, Massimiliano Caramia, and Pasquale Carotenuto. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies*, 10(1):19–34, 2002.
6. Simon Koblar and Luka Mladenovič. Calculating the speed of city bus trips: The case of Ljubljana, Slovenia. *Urbani izziv*, 31(1):112–122, 2020.
7. Fang Zhao and Ike Ubaka. Transit network optimization–minimizing transfers and optimizing route directness. *Journal of public transportation*, 7(1):63–82, 2004.
8. Zhongzhen Yang, Bin Yu, and Chuntian Cheng. A parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering*, 22(1):44–55, 2007.
9. Wei Fan and Randy B Machemehl. Using a simulated annealing algorithm to solve the transit route network design problem. *Journal of transportation engineering*, 132(2):122–132, 2006.
10. Faruk Serin and METE Süleyman. Public transportation graph: A graph theoretical model of public transportation network for efficient trip planning. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 25(4):468–472, 2019.

11. Peter A Steenbrink. Transport network optimization in the Dutch integral transportation study. *Transportation Research*, 8(1):11–27, 1974.
12. Google. Distance Matrix API. https://developers.google.com/maps/documentation/distance-matrix, Accessed: 2024-05-28.