MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Fišer Tomáš** Personal ID number: **420124**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Branch of study: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Algorithmic Analysis of Intermodal Transport Network

Master's thesis title in Czech:

Algoritmická analýza intermodální dopravní sítě

Guidelines:

Algorithmic analysis of transport networks aims to evaluate a set of key performance indicators that quantify how well the transport networks serves the mobility needs of its users. Evaluating such KPIs is an algorithmically demanding tasks requiring computing trips for millions of origin-destination pairs. The aim of this thesis is to develop algorithms for algorithmic analysis of intermodal transport networks involving many types of transport services and their combinations.
1. Survey existing methods for intermodal journey planning and transport network analysis
2. Formalise the intermodal transport network analysis problem with preferences of the city
3. Propose an algorithm to solve the intermodal transport network analysis problem with preferences
4. Implement the proposed algorithm
5. Evaluate the implemented algorithm on a real-world city-size intermodal transport network

Bibliography / sources:

[1] T. Pajor, D. Delling, R. F. Werneck. Round-based public transit routing. Society for Industrial and Applied Mathematics, 2012.
[2] T.Pajor. Algorithm Engineering for Realistic Journey Planning in Transportation Networks, PhD thesis, 2013.
[3] J. Nykl. Agent-based public transport network analysis. Bachelor's thesis, Czech Technical University in Prague, 2013.
[4] D. Delling, J. Dibbelt, T.Pajor. Fast and Exact Public Transit Routing with Restricted Pareto Sets. 10.1137/1.9781611975499.5, 2019.

Name and workplace of master's thesis supervisor:

Jan Hrnčíř, MSc., Ph.D., Department of Computer Science, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **19.09.2019** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until: **19.02.2021**

Jan Hrnčíř, MSc., Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

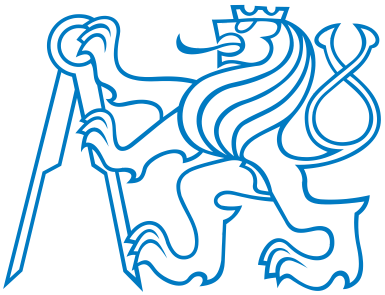
III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science



Master's Thesis

Algorithmic Analysis of Intermodal Transport Network

Bc. Tomáš Fišer

Supervisor: Jan Hrnčíř, Ph.D.

Study Programme: Open Informatics

Specialisation: Artificial Intelligence

May 22, 2020

Aknowledgements

At first, I would like to thank my supervisor Jan Hrnčíř for the excellent guidance during the whole process of this thesis. Also, I thank Michal Jakob, Jan Nykl, and Jan Hrnčíř for the feedback and providing me the opportunity to work on such a challenging topic.

In the end, this work would not be completed without access to computing machines in the Czech National Grid Infrastructure. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Declaration

I declare that the presented work was developed independently and I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague on May 22, 2020 .....

Abstract

This work focuses on the analysis of the intermodal transport network using a multi-criteria algorithm that considers preferences of the city. To perform the analysis, we first describe the representation of the intermodal transport network. Given the representation, we define the intermodal transport network analysis problem with preferences of the city. We aim at algorithmic analysis, which computes key performance indicators using given travel demand. Thus, we provide various key performance indicators, e.g., the number of overcrowded trip segments, the total duration of all passenger journeys, and the total costs of passenger journeys. The goal of the analysis is to optimize the number of overcrowded parts of the public transport network. To achieve the goal, we offer passengers alternative journeys. These journeys try to avoid public transport vehicles with occupancy beyond a certain level of comfort. In other words, a passenger may choose another public transport connection, ride a bike, or use a taxi service. We propose a multi-criteria algorithm that finds a suitable journey for each passenger while optimizing four criteria, i.e., vehicle occupancy, duration, costs, and the number of interchanges. We also implement an analysis tool that includes the multi-criteria algorithm and calculates the required key performance indicators. By using the analysis tool, we perform an analysis using the intermodal transport network of the capital city of Prague. In the evaluation, we achieve the reduction in the number of overcrowded trip segments in the intermodal transport network by 79.4 % on randomly generated travel demand.

Abstrakt

Tato práce je zaměřena na analýzu intermodální dopravní sítě pomocí multikriteriálního algoritmu s ohledem na priority města. Nejprve popisujeme reprezentaci intermodální dopravní sítě. Poté definujeme úlohu analýzy nad danou reprezentací. Jedná se o algoritmickou analýzu, tedy na základě zadané poptávky cestujících vyhodnocujeme klíčové indikátory. Mezi zahrnuté indikátory patří počet přeplněných úseků spojů, doba jízdy všech cestujících a celkové náklady všech cestujících. Cílem analýzy je optimalizovat počet přeplněných úseků dopravní sítě tím, že nabídneme cestujícím alternativní jízdy. Tyto cesty se snaží vyhnout úsekům dopravní sítě, kde jsou spoje přeplněné. Vyhnout se lze vybráním jiného spoje veřejné dopravy, jízdou na kole, nebo využitím taxi služby. Popisujeme multikriteriální al-

goritmus, který pro každého cestujícího vyhledá vhodnou cestu, přičemž optimalizuje čtyři kritéria: obsazenost vozu, dobu jízdy, cestovní náklady a počet přestupů. Také implementujeme nástroj pro analýzu, který obsahuje tento multikriteriální algoritmus a z nalezených cest vypočítá chtěné klíčové indikátory. Pomocí našeho nástroje provádíme analýzu intermodální dopravní sítě hlavního města Prahy. Při evaluaci námi vygenerované poptávky cestujících dosahujeme snížení počtu přeplněných úseků spojů v intermodální dopravní síti o 79,4 %.

Contents

Chapter 1

Introduction

Public transport is essential for citizens that need to move across a city on a daily basis, especially in highly populated cities, where citizens are facing peak hours. At first, in the morning when they are going to work or school, then in the evening when they travel back home. In these hours, public transport vehicles are usually crowded with passengers, which may be uncomfortable. Many people solve this problem by using a private car to travel. It is understandable because a lot of people feel more comfortable in their cars than in crowded public transport vehicles. Unfortunately, this behavior has consequences, e.g., a lack of free parking spaces or traffic congestion. According to the research made by INRIX[1], every London driver lost 149 hours on average in 2019 due to traffic congestion. In another study called the TomTom Traffic Index[2], Bengaluru in India is ranked as the most congested city in the world, where a journey in the evening takes an average of thirty minutes longer than in a free flow situation.

We think that the only way to make a highly populated city suitable for living is to provide a robust and well-designed *intermodal* transport network. There is another often occurred term in the literature, the *multimodal* transport network. The problem with these terms is that the definitions are not fully established, which leads to the fact that the terms are usually not differentiated. In many publications, using one of the terms simply means that multiple transport modes are included in the network. In our work, we decided to distinguish these terms, as it is described in [1]. On the one hand, the *multimodal transport network* describes a transport network, where only public transport modes are included, e.g., bus, tram, subway, train, and ferry. On the other hand, the *intermodal transport network* describes a transport network that combines transport modes for public transport, individual transport (e.g., car, bike, walk), and on-demand transport (e.g., taxi). We claim that such a definition of the intermodal transport network fully describes the real-world transport in a city.

As we would like to contribute to the improvement of transport in cities, we have decided to analyze the intermodal transport network. We aim at the algorithmic analysis that considers the preferences of the city. We assume that a city requires to avoid overcrowding in public transport vehicles. In the case travel demand is known, we should be able to

[1] http://inrix.com/scorecard/

[2] https://www.tomtom.com/en_gb/trafficindex/

detect the weak spots of the public transport network. Furthermore, if a city provides us a list of overcrowded segments in the transport network, we should be able to analyze how the crowdedness can be reduced. One option is to recommend alternative journeys to passengers who travel in crowded public transport vehicles. A passenger usually has a choice of several journeys. Although, the passenger often demands the fastest journey. We would like to assign a single alternative journey to each passenger where the increase of the journey's duration is not high in comparison to the fastest journey. Also, the journey's cost and the number of interchanges have to be considered during the journey selection. To be able to measure the effectiveness of the analysis, we define several *key performance indicators* (KPIs), e.g., the total duration of journeys, the total cost of journeys, and the number of overcrowded trip segments. Our primary goal is to describe and implement an analysis tool that takes travel demand as the input and returns KPIs as the output.

1.1 Structure of the Thesis

In Chapter 2, we begin with a research of existing publications where the topic of the intermodal journey planning and the transport network analysis is examined. We follow by formalizing the intermodal transport network analysis problem with preferences of the city in Chapter 3. Then, we provide a solution to the problem in Chapter 4. In Chapter 5, we describe our implementation of the proposed solution in detail. The analysis using the real-world data can be found in Chapter 6. Finally, we summarize our work and discuss future extensions in Chapter 7.

Chapter 2

Related Work

In this chapter, we survey recent studies related to the topic of this thesis. Firstly, we present list publications that aim at the analysis in the transport network. Then, we present selected publications with solutions of journey planning problems. Finally, we describe our contributions to the topic of intermodal transport network analysis.

2.1 Transport Network Analysis

Many ways of analyzing the transport network were invented in the last years. Generally, the analysis is based on the specification and computation of key performance indicators (KPIs). Usually, the transport network is modeled as a graph. Thus the topological indicators are mostly presented, e.g., node degree or betweenness centrality.

- **Node degree:** A metric for a node in the graph which is computed as the number of nodes connected by an edge with the given node.
- **Betweenness centrality:** A metric for a node (or an edge) in the graph computed as the number of shortest paths that traverse the given node (or the given edge).

There are many more metrics used in the literature. Since we do not describe them in detail, we recommend the survey [14] with a list of all standard metrics used for transport network analysis.

2.1.1 Level of Service

The interesting methodology for analyzing the performance of the intermodal transport network is described in [2]. The work is aimed at the computation of the multimodal performance index (MPI) that measures the quality of the transport network in which walk, bicycle, and public transport modes are included. The MPI is a single value computed by aggregating mode-specific key performance indicators (KPIs). The mode-specific KPIs are computed separately for every transport mode. Members of these KPIs are person-related *delay* and *level of service*.

- **Delay:** A difference between actual travel time and the minimum travel time.
- **Level of service:** A classification of the traffic flow into six levels, rated from A (best) to F (worst).

In [3], the multimodal level of service is deeply studied with an entirely different definition. The authors also determine the level of service by six grades from A to F. The difference is that the authors use a *quality of service* to classify the level of service (in [2], the level of service is determined by the traffic flow). The quality of service is a measure based on the travel experience of the users (and potential users) of the specific transport mode, e.g., bicycle, walk, or public transport modes.

2.1.2 Network Modeling from Geographical Data

To perform an analysis on the transport network, it is crucial to model the network from real-world geographical data. For this purpose, in [6], the author describes how to build multimodal network models in order to analyze them. To create the model, the author uses the *OpenStreetMap*[1] (OSM) data and other volunteered geographic information. The work contains valuable information about OSM data quality and availability. The spatial data model design combines particular parts of the transport network, e.g., private transport network and public transport network.

2.1.3 Non-public Transport Network Analysis

The non-public transport networks are analyzed in the following studies. In [7], the authors claim that characterizing the transport network purely on topological means is insufficient. Therefore, the authors include additional measures, namely the fastest-path interpolation and the routing-service interpolation. Then, the authors propose the correlation analysis of two transport networks based on GPS traces of taxis in San Francisco and Shanghai. Then, they identify the driving behavior during peak hour and non-peak hours. The authors also prove that including the node weight and the travel speed into the betweenness analysis improves the traffic prediction performance. The similar attempt is in [8], where the authors also use GPS traces of taxis to estimate the traffic flow in Qingdao. The interesting point is that the authors use the Monte Carlo algorithm to simulate the traffic distribution. Then, they measure the correlation between the simulated and observed data.

2.1.4 Public Transport Network Analysis

The analysis of the public transport network is thoroughly studied in many publications. The authors in [9] model a graph of the network and the traffic flows using the timetables. Their representation is formed by space-of-changes, space-of-stations, and space-of-stops. In the analysis, they measure a node load by a weighted combination of four metrics: simple load, node degree, betweenness, and restricted betweenness. The authors compare three large transport networks in Warsaw, Switzerland, and the whole of Europe. They conclude that the real-world traffic patterns are significantly heterogeneous in traffic flow intensity.

[1] https://openstreetmap.org/

Another approach is used in [10], where the authors propose a network model called the *supernode graph structure representation*. A supernode is a set of nodes that are geographically close to each other. The bus and metro transport networks are both represented using the graphs of supernodes. Then, the *spatial amalgamation* method is used to combine these two transport layers. Then, the analysis is based on assigning weights to nodes in the graph. They evaluate their method on the bus and metro transport network of London by comparing the mono-layer and the proposed multi-layer representation. Their finding is that analyzing separated mono-layers may lead to significantly different network behavior than real-world usage is.

In [11], the authors model the network as a graph with weighted directed edges, where the weight represents the capacity of the bus, tram or trolleybus in the peak morning hour. The authors measure 11 metrics, including average path length, eccentricity distribution, and betweenness centrality. One interesting metric they use is page rank centrality. The idea behind page rank centrality is to differentiate nodes with the same in-degree. The *PageRank* [12], originally used in a web search engine, is a scoring algorithm that iteratively assigns the PageRank value to each node in the network, where a node with the incoming edge from the strongly linked node is considered as more important. In the evaluation, the authors analyze the public transport network of five Hungarian cities, and they identify which routes and stations of the public transport network are overloaded.

The structural efficiency of the metro transport network from the network science perspective is studied in [13]. In the work, two criteria are optimized, number of transfers and distance. The authors propose a metric called *node occupying probability* to measure the level of utilization of stations. The analysis is based on *random failure* and *target attack* in metro network. The random failure is simulated by the random deletion of several nodes. To simulate the target attack, the nodes with the highest betweenness or with the highest node occupying probability are deleted. According to the analysis of six metro networks (Beijing, London, Paris, Hong Kong, Tokyo, and New York), the New York metro has the best topological efficiency. Hong Kong's metro is most robust under random attack, and Tokyo is the least vulnerable to the target attack.

In [4], the author implemented a tool that analyzes the public transport network by computing eight KPIs, including node degree, trip workload, and betweenness centrality. The tool is able to calculate all metrics within 10 ms per query.

A very comprehensive analysis is provided in [5]; the authors compare public transport networks of fourteen cities. They measure many metrics on four types of graphs for each transport network. During the analysis, the authors explain many counter-intuitive observations, e.g., why routes share the same path instead of exploring more areas.

Finally, a well-written survey on recent work in public transport analysis is [14]. The survey summarizes various graph models and topological characteristics used in other studies.

2.1.5 Applications

The European Cooperation in Science and Technology[2] (COST) organized an action called Accessibility Instruments for Planning Practice in Europe. During this action, 24 instru-

[2]https://www.cost.eu/who-we-are/about-cost/

ments were reviewed and compared. One of the instruments is the Spatial Network Analysis for Multimodal Urban Transport Systems (SNAMUTS), firstly provided in [15], [16]. The SNAMUTS is able to compute the following KPIs for each node:

- **Closeness centrality:** The minimum cumulative impediment value between each pair of nodes.
- **Degree centrality:** The minimum number of transfers between each pair of nodes.
- **Betweenness centrality:** The geographical distribution of attractive travel paths between each pair of nodes.
- **Contour catchment:** The number of residents and jobs within the walkable catchment areas of nodes that can be reached within a public transport up to 30 minutes from the reference node.
- **Nodal connectivity:** The suitability of nodes for making transfers with minimal disruption to the flow of movement.
- **Nodal resilience:** The overcrowding index of a node.
- **Composite index:** The above metrics combined into a single value for each node.

More details about metrics are available on the SNAMUT's website[3] as well as the result of computation for selected cities.

Another instrument studied during the COST's action is the Walk Score[4]. The web-based application provides several scoring techniques to measure the quality of a location in the transport network. The following metrics are included:

- **Walk score:** The metric that analyzes walking routes from the location to nearby amenities. For every amenity within 30 minutes away, points are added to the location. The score determines how a location is independent of a car or another way of transport.
- **Transit score:** The metric that describes how well the location is served by public transport. The Transit score is based on measuring the distance, frequency, and type of nearest routes.
- **Bike score:** The metric that describes how the surrounding area of the location is suitable for biking.

Every score values are in the range from 0 to 100. The authors even patented the walk score and the transit score metrics.

[3] http://www.snamuts.com/
[4] https://www.walkscore.com/

2.2 Journey Planning Algorithms

The journey planning topic is actively studied for many years, and plenty of works were published. We will focus on publications that provide algorithmic solutions to journey planning problems. Generally speaking, the *journey planning* (or route planning) is a group of problems where the common problem is to find an optimal journey (or multiple journeys) in a transport network. In publications, the representations of the journey or transport network vary. The most common representation comes from graph theory. The transport network is a *directed weighted graph*, and the journey is a path. The *path* is a sequence of nodes in the graph where the first node in the sequence is called *origin*, and the last node is called *destination*. The *journey planning algorithm* is an algorithm that solves a certain journey planning problem. There are almost no limits in specifying the criterion which is optimized in journey planning problems. The most occurred criteria are journey duration or distance. A more critical property is whether the journey planning problem requires the optimization of a single criterion or multiple criteria at once. The problems where multiple criteria have to be optimized are more challenging and result in a full *Pareto set* of non-dominated journeys.

Very exhausting surveys on journey planning algorithms of any kind are [1], [17], [18], and [19]. In the following text, we categorize the journey planning methods by the type of transport network in which they operate and by the number of optimized criteria.

2.2.1 Journey Planning in Road Network

The *Road network* is the transport network of roads where the non-public transport modes (car, bicycle, taxi service, etc.) operate. In journey planning, the Road network is mostly represented as a directed weighted graph. Nodes are junctions and edges are the roads between them.

Single criterion algorithms: The basic algorithm that finds the shortest path in a graph is Dijkstra's algorithm [20]. The Dijkstra's algorithm is not applicable to solve the problem in large road networks since its computational complexity is $\mathcal{O}(|E| + |V| + log|V|)$ (when the Fibonacci heap is used). Fortunately, the algorithms that solves the problem in microseconds on large graphs were invented, e.g., Contraction Hierarchies [21], Customizable Route Planning [22], [23], SHARC [24], or ALT [25]. Contraction Hierarchies and Customizable Route Planning have one property in common. The algorithms are divided into the query phase and the pre-processing phase. The graph is pre-processed once, but the computation is demanding on time. Then, the query phase computes the journeys quickly. The consequence is that when the graph is modified, the pre-processing phase needs to be redone. In [26], the author presented the External Customizable Route Planning, which is a variant of the Customizable Route Planning that does the pre-processing phase within few minutes in the external memory of the mobile device.

Multi-criteria algorithms: Finding the Pareto set of journeys is computationally more expensive. The computation time of the algorithm grows with the size of the resulting Pareto set. One of the algorithms that solve the multi-criteria problem is called multi-criteria label-setting algorithm (MLS) [27]. MLS is an extension of Dijkstra's algorithm that stores a bag

of non-dominated labels for each expanded node. Each label has a vector of criteria values, and one label dominates the other if it has a strictly better value of at least one criterion, and it does not have worse value in other criteria. In comparison to Dijkstra's algorithm, the priority queue stores labels in lexicographic order. In each iteration, the minimal label in the priority queue is expanded. Another approach is the multi-criteria label-correcting algorithm (MLC) [28]. The main difference from MLS is that all non-dominated labels of the current node are processed at once. It leads to visiting the same label more times.

Instead of extending Dijkstra's algorithm, the MOA* [29] and NAMOA* [30] algorithms extends the A* algorithm [31]. These algorithms use an additional search graph and heuristic vector. In [32], the authors proposed a heuristic for NAMOA* that speed-up the bi-criteria road routing [33] in order of magnitude.

2.2.2 Journey Planning in Public Transport Network

For journey planning in the public transport network, it is specific that transport modes are limited by timetables. Also, the network usually contains multiple public transport modes (e.g., bus, tram, subway, train, ferry). The intermodal transport network is usually modeled as time-dependent graph [34] or time-expanded graph [35]. On the one hand, the time-dependent graph has one node for each stop. On the other hand, the time-expanded graph has defined nodes by events at stops, e.g., departure time of public transport vehicles. One exception is [36], where the model of the public transport network are lists of routes and trips.

Single criterion algorithms: Here, we consider the journey problems where one criterion is optimized. If the public transport network contains only one public transport mode, the algorithms for road networks may be applied, e.g., SHARC or Contraction Hierarchies. More often are journey planning problems with more public transport modes. For that purpose, several graph-based algorithms were proposed, e.g., Layered Dijkstra [34] or Public Transit Labeling [37]. Nevertheless in [36], the authors invented the RAPTOR algorithm that is not graph-based. Since the public transport network contains scheduled trips and one trip is a defined sequence of stops, the authors prove that it is better to use a different approach that searches through trips round by round from a starting stop. The authors claim that the computation time of the RAPTOR algorithm is faster in order of magnitude than previous approaches using the time-dependent or time-expanded model. Furthermore, the RAPTOR algorithm optimizes travel time and the number of interchanges by default. That means the output of the algorithm is a full Pareto set of non-dominated journeys.

Multi-criteria algorithms: An attempt to minimize three criteria (travel time, walking time, number of transfers) in the public transport network is in [38]. The authors model a graph for each transport mode where nodes are stations, platforms, and departure events. Then the particular graphs are integrated into one larger graph. Then a modification of Dijkstra's algorithm is used to find the Pareto set. In [39], the authors provide the gradual approach to minimize travel time and the number of interchanges. The network is represented by the time-dependent graph with shortcuts, which is daily recomputed using data provided by transport institutions. The query phase searches k journeys in rounds.

At first, journeys without transfer are examined; if there is no k journeys returned, the algorithm continues to search journeys with one transfer. This is repeated until the search with at most five transfers is done. The particular searches are based on a modification of Dijkstra's algorithm, and the authors claim that the approach is also inspired by Contraction Hierarchies. The authors evaluate the algorithm, called the Gradual Path-finding algorithm, on the public transport network of Izmir city. They observe that the algorithm processes the query in 0.63 seconds on average. The authors of the RAPTOR algorithm [36] propose in the same publication a modification of the RAPTOR algorithm, which is able to consider additional criteria (McRAPTOR). Unfortunately, the performance of the McRAPTOR is not good enough for real-time applications. Thus, partly the same group of authors improve the McRAPTOR algorithm in [40]. They define the *Restricted Pareto Set*, which is the subset of Pareto optimal journeys where the journeys that are considered as not useful in the real-world are omitted. The authors propose an algorithm that is composed of three phases wherein each phase a different modification of the RAPTOR algorithm is used. They use the Fuzzy dominance scoring algorithm [41] as the tool to score the journeys in the Pareto set. By using the scoring algorithm, they observe that the best-rated journeys in the Pareto Set are also in the Restricted Pareto Set.

2.2.3 Journey Planning in Intermodal Transport Network

As we mentioned in Chapter 1, the intermodal transport network intersects the public transport network and the road network. Hence, the journey planning problem is to find a journey that may be a combination of scheduled public transport modes (e.g., bus, tram, subway, train, ferry) and individual transport modes (e.g., bike, walk) and on-demand transport modes (e.g., taxi service).

In [42], the authors combine two approaches to provide an algorithm for the intermodal transportation network. A modification of the RAPTOR algorithm is used for planning on the public transport network, and the Dijkstra-based algorithm is used for planning on other transport modes. In [43], the author presented several versions of the speed-up algorithm called State Dependent ALT (SDALT). The SDALT algorithm is an extension of ALT [25] that also performs well on the intermodal transport networks.

2.3 Our Contribution

While a lot of studies were presented in the field of journey planning or transport network analysis, we were not able to find a work that solves the intermodal transport network problem with preferences of the city. By preferences of the city, we mean that a city has the data about travel demand and overcrowded parts in the transport network and wants to distribute the demand in a way that overcrowding in public transport vehicles is reduced. In this thesis, we use the existing journey planning algorithms to form the Intermodal CRP-McRAPTOR algorithm that analyzes the intermodal transport network with preferences of the city. The Intermodal CRP-McRAPTOR algorithm distributes the given travel demand in the intermodal transport network by searching optimal journeys that avoid to the highly occupied public transport vehicles. In order to measure the vehicle occupancy, we also define several KPIs, e.g., number of overcrowded segments and total duration of journeys.

Chapter 3

Problem Specification

In this chapter, we describe our representation of the intermodal transport network. Then, we provide definitions of travel demand and preferences of the city. We also specify the key performance indicators (KPIs) that are used in this thesis to analyze the intermodal transport network. Finally, we formalize the intermodal transport network analysis problem with preferences of the city.

3.1 Transport Modes

Since we analyze the intermodal transport network, we need to consider standard public transport modes, individual transport modes, and on-demand transport modes. We use the following notation:

- **Subway (S):** Public transport mode where a *subway* is used to travel. The maximum capacity of the subway $c^s \in \mathbb{N}^+$ is defined as the maximum number of passengers that one subway train is able to transport between two stops.
- **Bus (B):** Public transport mode where a *bus* is used to travel. The maximum capacity of the subway $c^b \in \mathbb{N}^+$ is defined as the maximum number of passengers that one bus is able to transport between two stops.
- **Tram (T):** Public transport mode where a *tram* is used to travel. The maximum capacity of the subway $c^t \in \mathbb{N}^+$ is defined as the maximum number of passengers that one tram is able to transport between two stops.
- **Rail (R):** Public transport mode where a *city train* is used to travel. The maximum capacity of the subway $c^r \in \mathbb{N}^+$ is defined as the maximum number of passengers that one train is able to transport between two stops.
- **Ferry (F):** Public transport mode where a *ferry* is used to travel. The maximum capacity of the subway $c^f \in \mathbb{N}^+$ is defined as the maximum number of passengers that one ferry is able to transport between two stops.
- **Public Transport (PT):** A group that includes all above public transport modes.

- **Walk (W):** Individual transport mode where a *walk* is used to travel.
- **Bicycle (BC):** Individual transport mode where a *bicycle* is used to travel.
- **Taxi Service (TS):** On-demand transport mode where taxi service is used to travel.
- **Non-public Transport (NPT):** A group that includes individual transport modes (W, BC) and on-demand transport modes (TS).

The capital letters are used to simplify the full name of the transport mode or its group. Every public transport mode may contain multiple vehicles that operate during a day. Also, the maximum capacity of a vehicle is defined only for public transport modes. For W, BC, and TS modes, the maximum capacity is not defined, since it's considered that passenger travels alone.

To represent all transport modes included in the model of the intermodal transport network, let us define a set of *supported transport modes* $\mathcal{M}$ as a set that contains all available modes in the intermodal transport network. Also, we want to observe how KPIs differ when we add or remove any transport mode from the intermodal transport network. Thus, we need a set that includes only selected transport modes from $\mathcal{M}$. Formally, we define a set of *allowed transport modes* as a set $\mathcal{M}_a \subseteq \mathcal{M}$. In a single process of analysis, only allowed transport modes $\mathcal{M}_a$ may be used by passengers.

3.2 Road Network Representation

From our perspective, the *road network* of the city is a system of roads where citizens travel by individual or on-demand transport modes. To represent a road network, we use a directed multi-weighted graph called the *Road Graph*. The Road Graph contains roads and crossroads with relation to its geographical locations. Formally, we define the *Road Graph* as a directed weighted graph $G_r = (V_r, E_r, \delta, \nu, R_r, \varrho)$, where

- V_r is a set of vertices describing the junctions where a vertex $v \in V_r$ is defined by its latitude $lat_v \in \mathbb{R}$, longitude $lon_v \in \mathbb{R}$, and elevation $elev_v \in \mathbb{R}$,
- E_r is a set of edges describing roads between junctions,
- a function $\delta : E_r \mapsto \mathbb{R}_0^+$ assigns a distance for each edge $e \in E_r$,
- a function $\nu : E_r \mapsto \mathbb{R}_0^+$ assigns an average speed for each edge $e \in E_r$,
- R_r is a set of all road categories that occur in the graph,
- a function $\varrho : E_r \mapsto R_r$ assigns a road category for each edge $e \in E_r$.

3.3 Public Transport Network Representation

The *public transport network* of the city is a transport system that includes roads and scheduled trips of public transport modes, e.g., B, S, T or R. We represent the public transport network the same way as in [36]; the *Timetable* is a tuple $\Upsilon = (\Pi, \mathcal{S}, \mathcal{T}, \mathcal{R}, \mathcal{F})$ where

- $\Pi \subset \mathbb{N}$ is the period of operation, e.g., seconds of the day,
- $\mathcal{S}$ is a set of stops, and a stop $s \in \mathcal{S}$ corresponds to a location in the network where a person can board or get off a vehicle,
- $\mathcal{T}$ is a set of trips, and a trip $t \in \mathcal{T}$ represents a sequence of stops a vehicle visits along a route and each stop in the trip has its arrival time $\tau_{arr} \in \Pi$ and departure time $\tau_{dep} \in \Pi$ for which holds $\tau_{arr} \leq \tau_{dep}$,
- $\mathcal{R}$ is a set of routes, and a route $r \in \mathcal{R}$ consists of a set of trips that shares the same sequence of stops,
- $\mathcal{F}$ is a set of transfers, and a transfer $f \in \mathcal{F}$ consists of two stops p_{from}, p_{to} and the transfer time $l(p_{from}, p_{to}) \in \mathbb{R}^+$ between two stops.

3.4 Intermodal Transport Network Representation

As we described in Chapter 1, the *intermodal transport network* in the city is a complete system of roads where citizens may travel by a combination of transport modes from PT and NPT. Thus, the journey is divided into a sequence of *legs* where a *leg* is a part of the journey determined by a certain transport mode. We represent the intermodal transport network as a tuple $\mathcal{I} = (\mathcal{M}, G_r, \Upsilon)$, where $\mathcal{M}$ is a set of supported transport modes defined in Section 3.1, G_r is a road graph defined in Section 3.2 and Υ is a timetable from Section 3.3.

3.5 Preferences of the City

We know that everyone has a different point of view on what the city should demand from the transport system, e.g., low-cost tickets for citizens, quality of service, etc. In our opinion, the overcrowding in public transport vehicles decreases quality of living in the city. Therefore, the city should require the transport system that prevents overcrowding in public transport vehicles. More formally, let us have a tuple (E_s, φ) where E_s is a set of all trip segments in the public transport network, and $\varphi : E_s \mapsto \mathbb{R}$ is a function that assigns a *level of occupancy* to each trip segment in E_s. A *trip segment* is defined by two stops, scheduled trip and public transport mode associated with the trip. The higher value assigned by the function φ determines that the trip segment is more occupied. Then, we say that the city prefers to spread out passengers from highly occupied segments into surrounding less occupied segments. As a result, the total overcrowding in public transport vehicles has to be decreased.

3.6 Travel Demand

To reduce the overcrowding in public transport vehicles, we need to know how many people are using the intermodal transport network at a time. In other words, we need from every traveling citizen a starting location, destination location, and a time of departure. Formally, we define the *travel demand* $\mathcal{D}$ as a tuple (L, τ) where

- L is a set of origin-destination pairs,
- τ is a function that assigns a departure date-time to every origin-destination pair from L.

An origin-destination pair $l \in L$ is composed of the origin location l_o and the destination location l_d. Therefore, the size of the L is equal to the number of citizens in the travel demand.

3.7 Key Performance Indicators

Key performance indicators (KPIs) are a set of the leading indicators that measure how effective the progress towards the desired result is. In an analysis, this usually means that we have a set of target values that we compute for each scenario. Then, these values are compared to determine which of the scenarios is closest to the desired objectives of the analysis. In this thesis, we focus on dynamic metrics that measure the movement of citizens in the intermodal transport network. Hence, we search journeys for origin-destination pairs and aggregate their parameters into network-based KPIs. For these metrics, four criteria are essential, vehicle occupancy, duration, costs, and the number of interchanges. We define a set of KPIs in the intermodal transport network as the set K that includes the following KPIs.

- k_n: Number of journeys
- k_{dt}: Total duration of journeys
- k_{ct}: Total costs of journeys
- $\overline{k_d}$: Average journey duration
- $\overline{k_c}$: Average journey costs
- $\overline{k_{noi}}$: Average number of interchanges
- k_u: Number of used trip segments
- k_i: Number of inconvenient trip segments
- k_o: Number of overcrowded trip segments
- k_{su}: Number of overcapacity trip segment usages

In the rest of this section, we describe the particular KPIs in detail.

3.7.1 Feasibility

For several reasons, no journey may be found for an origin-destination pair. A passenger may have requested a journey with a departure time for which no transport service is available. Another possibility is that the origin or destination is too far from the transport network. To observe these situations, we define the KPI $k_f \in \mathbb{N}$ called *number of journeys* as the number of origin-destination pairs for which at least one feasible journey is found.

3.7.2 Journey Duration

An indicator that measures a time that passengers spent by traveling must not be left out. Many citizens want to travel as fast as possible, especially those that live in the city with a high population density. Let us define a *duration* $d(j) \in \mathbb{N}$ of a journey j as a duration (in seconds) between the departure at origin location and the arrival to the destination location. Then, the KPI $k_{dt} \in \mathbb{N}$ called *total duration of journeys* is computed as follows

$$k_{dt} = \sum_{j \in J} d(j)$$

where J is a set of journeys and $d(j)$ is a duration of a particular journey $j \in J$. For each origin-destination pair in the travel demand, there is at most one journey in J. Then, we define the KPI $\overline{k_d} \in \mathbb{R}$ called *average journey duration* by the following equation:

$$\overline{k_d} = \frac{\sum\limits_{j \in J} d(j)}{|J|}$$

The J and $d(j)$ are the set of journeys and journey duration, respectively. In other words, the $\overline{k_d}$ describes the average duration (in seconds) per one passenger.

3.7.3 Journey Costs

Similarly, we define *costs* $c(j)$ of journey j as costs that a passenger need to pay (in Euro) for the realization of the journey. Then, given a set of journeys J and journey costs $c(j)$, the KPI $k_{ct} \in \mathbb{N}$ called *total costs of journeys* is defined as follows

$$k_{ct} = \sum_{j \in J} c(j)$$

and the KPI $\overline{k_c} \in \mathbb{R}$ called the *average journey costs* is defined by the following equation:

$$\overline{k_c} = \frac{\sum\limits_{j \in J} c(j)}{|J|}$$

Also, the average journey costs are in Euro and describes how much a passenger needs to pay for a journey on average.

3.7.4 Number of Interchanges

The third criterion that we consider is the number of interchanges in a journey. This criterion is sometimes more important to passengers than the duration. When a journey with no interchange is a few minutes slower than the fastest journey with one interchange, a passenger may choose the slower journey since it is more comfortable. Let us define the *number of interchanges* $noi(j)$ in a journey j as the number of transfers made by a passenger during the journey j. To include this criterion into KPIs, we define the KPI $\overline{k_{noi}} \in \mathbb{R}$ called *average number of interchanges* as follows:

$$\overline{k_{noi}} = \frac{\sum\limits_{j \in J} noi(j)}{|J|}$$

In the equation, the J is a set of journeys and $noi(j)$ is the number of interchanges in the particular journey $j \in J$.

3.7.5 Occupancy

As we defined in Section 3.5, the city prefers to spread out the passengers from highly occupied segments in the intermodal transport network. Let the E_s be a set of trip segments. The *trip segment* $s \in E_s$ is defined by two stops, trip, and the transport mode that is assigned to the trip. Then, the trip segment is *used by passenger* if the passenger's journey contains the trip segment. The *occupancy* of the trip segment $u(s) \in \mathbb{N}$ is defined by the number of passengers that use the trip segment to travel. At first, we define KPI $k_u \in \mathbb{N}$ called the *number of used trip segments*, which is the count of trip segments that are used by at least one passenger. The next KPI $k_o \in \mathbb{N}$ is the *number of overcrowded trip segments*. The k_o is a count of trip segments that are occupied above the maximum capacity c_{max} of the transport mode on the segment.

Also, we want to be able to observe trip segments that are almost fully occupied. Thus, before defining a new KPI, we need to introduce the *maximum convenient occupancy* $b_{conv} \in \mathbb{R}$ of the trip segment, and a *convenience function* $\varsigma : E_s \mapsto \{0, 1\}$. The maximum convenient occupancy determines the boundary of how many passengers may be in the public transport vehicle to consider the occupancy as comfortable. The function ς determines if the trip segment $s \in E_s$ is *convenient* (1) or *inconvenient* (0) by the following rule

$$\varsigma(s) = \begin{cases} 1 & \text{if } u(s) \leq b_{conv} \\ 0 & \text{otherwise} \end{cases}$$

where $u(s)$ is the occupancy of the trip segment $s \in E_s$. The difference of the convenient, inconvenient and overcrowded segments is depicted in figure 3.1.

We use the convenience function ς to define the KPI $k_i \in \mathbb{N}$ called the *number of inconvenient trip segments*. Similarly to k_o, the k_i is defined as the count of trip segments that are occupied above the maximum convenient occupancy b_{conv} of a transport mode that is assigned to the trip segment.

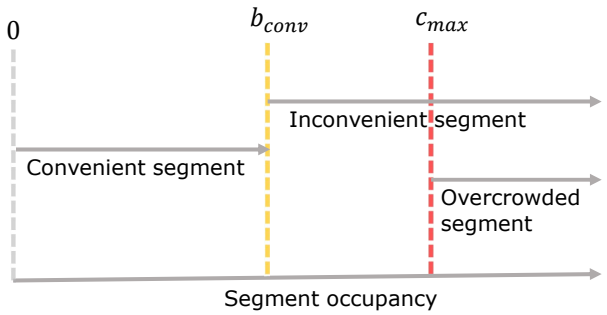


Figure 3.1: The schema of trip segment occupancy. The occupancy increases from the left to the right. The segments that are occupied above the convenient boundary b_{conv} are called *inconvenient* and the segments that are above the maximum capacity c_{max} are called *overcrowded*.

The last occupancy-based KPI $k_{su} \in \mathbb{N}$ is called the *number of overcapacity trip segment usages*. We assume that the *trip segment usage* is a passenger traveling across the trip segment. A trip segment usage is called the *overcapacity trip segment usage* if the passenger use the trip segment that is already fully occupied to the maximum capacity of the transport mode. In other words, if the maximum capacity of the transport mode is five and the occupancy on the trip segment is seven, then the number of overcapacity trip segment usages is two. The k_{su} is the number of overcapacity segment usages in all trip segments E_s. Denote that the k_{su} describes how many passengers would not fit into a public transport vehicle in each segment.

3.8 Intermodal Transport Network Analysis Problem with Preferences of the City

Finally, we have all set up to define the analysis problem studied in this thesis. By using the properties described in this chapter, the following parameters are given:

- intermodal transport network $\mathcal{I}$,
- set of allowed transport modes $\mathcal{M}_a$,
- preferences of the city (E_s, φ),
- travel demand $\mathcal{D} = (L, \tau)$.

Then, the *Intermodal Transport Network Analysis Problem with Preferences of the City* (ITNAP-PC) is to compute a set of KPIs K defined in Chapter 3.7. During the computation, only the modes in the $\mathcal{M}_a$ are allowed to be used by a passenger. By using this definition, we are able to observe the changes of KPIs for various settings of the input parameters.

Chapter 4

Solution Approach

In this chapter, we introduce our approach for solving the Intermodal Transport Network Analysis Problem with Preferences of the City (ITNAP-PC). The solution of ITNAP-PC is divided into the following steps. At first, we use the customizable route planning (CRP) technique for searching first mile journeys in the Road Graph G_r. Then, we convert the journeys from the first step to initial transfers that are accepted by a multi-criteria algorithm called McRAPTOR. The McRAPTOR algorithm returns a Pareto set of optimal journeys for each origin-destination pair. At this time, journeys in Pareto sets may be composed of parts of different transport modes. After we search all Pareto sets, we compute key performance indicators (KPIs) of the intermodal transport network as they are defined in Chapter 3. By merging above algorithms together we get the *Intermodal CRP-McRAPTOR algorithm* that solves the whole ITNAP-PC.

4.1 Customizable Route Planning

The *Customizable Route Planning* (CRP) in [22] solves a part of our problem, which is the journey planning in the Road Graph. The CRP is a technique for computation of journeys in road networks that focuses on customization for arbitrary criteria. The technique is divided into three phases:

1. **Metric-independent pre-processing:** Given a topology of the graph G_r, it finds multilevel partitions by dividing the G_r into nested cells on a fixed number of levels.

2. **Metric-dependent customization:** This phase builds a multilevel overlay from the graph partitions for a given metric. It stores the customization data independently to support multiple metrics for the same road network at the same time.

3. **Query phase:** At first, start and destination nodes are added to the pre-processed graph; it generates the search graph G'. Then, an algorithm searches for journeys in the search graph with respect to the given criterion. The algorithm uses the output from previous phases to speed-up the computation. For this phase, the Multi-level Dijkstra algorithm is usually used.

In our thesis, we use CRP to find first mile journeys from the origin locations to surrounding stops of the public transport network. For each non-public transport mode and origin location, we do the following. Firstly, we find the neighboring stops of the origin location within some range where each transport mode may have a different range defined. Then, we use CRP to search the optimal journey in the Road graph from the origin location to every stop within the range. The CRP algorithm optimizes single criterion. In this thesis, we decided to optimize the journey duration. Thus, we obtain a set of journeys for each origin location. The set includes journeys of all allowed non-public transport modes.

4.2 Contraction Hierarchies

In [21], the authors present the *Contraction Hierarchies* (CH) algorithm, which is the speed-up technique for finding optimal journeys in a graph. The algorithm is based on contracting nodes in the pre-processing phase and bidirectional search in the query phase.

1. **Pre-processing phase:** Nodes in the graph are sorted by importance, and then the least important node is replaced with shortcut edges. This whole process is repeated many times.

2. **Query phase:** A query is processed by the bidirectional search. The forward search expands the edges that direct to a more important node only. On the other hand, the backward search expands only the outgoing edges of the important node.

Since we need to process many one-to-all queries on the Road Graph, we use CH to speed-up the process of finding journeys with non-public transport modes.

4.3 McRAPTOR Algorithm

The multi-criteria version of the *Round-Based Public Transit Router* (McRAPTOR) search journeys for public transport modes given a timetable $\Upsilon = (\Pi, \mathcal{S}, \mathcal{T}, \mathcal{R}, \mathcal{F})$. It is introduced in [36], where the pseudo-code of the single criterion RAPTOR is also included. The main idea of the algorithm is to work in rounds where at most K rounds are processed. The algorithm starts by marking the departure stop p_s. A round k computes the fastest way of getting to every stop with at most $k-1$ transfers. During each round, the stops that were visited for the first time or with better arrival time are marked. In the following round $k+1$, the algorithm processes all marked stops. The algorithm could be stopped earlier at the end of a round if the terminal stop p_t was visited. The basic RAPTOR optimizes the arrival time and the number of interchanges. The McRAPTOR also optimize the arrival time and number of interchanges by default. Nevertheless, it is possible to add more criteria, since the McRAPTOR uses label bags to keep all non-dominated labels of a stop.

In this thesis, we use the McRAPTOR to solve the second part of our problem, which is journey planning in the public transport network. If we find the nearest stop for each origin and each destination, we can search for journeys that use strictly public transport modes, e.g., bus, tram, rail. As the criteria, we set the duration, occupancy, costs, and number of interchanges. Thus, the result of the McRAPTOR is a Pareto set of optimal journeys for each origin-destination pair where each journey may contain more public transport modes.

4.4 Fuzzy Dominance Scoring Algorithm

Since we work with Pareto sets, it is useful to be able to determine which journey is the most suitable for a passenger. For this purpose, the *fuzzy dominance scoring algorithm* is presented in [41], [42]. This algorithm uses properties from Fuzzy logic to score a journey from Pareto set in the range from 0 to 1.

Fuzzy logic is a generalization of propositional logic, where the main difference is in the assignment of truth values to variables. On the one hand, the values of variables in propositional logic are 0 (false) or 1 (true). On the other hand, the truth values in Fuzzy logic are real numbers from the inclusive interval [0,1]. The truth value assigned to a variable determines the *partial truth*, where 0 is completely false, and 1 is completely true.

The basic *dominance* is a relation between two journeys, where a journey j_1 dominates a journey j_2 if j_1 is strictly better than j_2 in at least one criterion, and no worse in any other criteria. Then, we say that the Pareto set is a set of journeys where every two journeys do not dominate each other. It does not hold for the *Fuzzy dominance*. The *Fuzzy dominance* gives more information about the relation between two journeys than the basic *dominance*. Loosely speaking, if we compare two journeys from the Pareto set by using the Fuzzy dominance, instead of obtaining a single value (true or false), we get a real number that determines how much a journey dominates another.

More formally, when two journeys (j_1, j_2) in Pareto set are compared by the Fuzzy dominance, we get a *degree of domination* $d(j_1, j_2) \in [0, 1]$. The *degree of domination* measures how much the first compared journey j_1 fuzzy-dominates the other journey j_2. The degree of domination is computed by the following equation:

$$d(j_1, j_2) = \begin{cases} \frac{2n_b(j_1,j_2),+n_e(j_1,j_2)-|M|}{n_b(j_1,j_2)} & \text{if } n_b(j_1, j_2) > \frac{|M|-n_e(j_1,j_2)}{2} \\ 0 & \text{otherwise} \end{cases}$$

where M is set of optimized criteria, $n_b(j_1, j_2)$ is fuzzy number of criterion in which j_1 is better than j_2. Similarly, the $n_e(j_1, j_2)$ is fuzzy number of criterion in which the j_1 is equally good as the j_2. The following equation applies for $n_b(j_1, j_2)$ and $n_e(j_1, j_2)$, respectively.

$$n_b(j_1, j_2) = \sum_{m \in M} \mu^m_<(k^m(j_1), k^m(j_2))$$

$$n_e(j_1, j_2) = \sum_{m \in M} \mu^m_=(k^m(j_1), k^m(j_2))$$

The $k^m(j_i)$ is a value computed for the journey j_i according to the criteria m, e.g. journey duration in seconds from Section 3.7.2. The $\mu^m_<$ and $\mu^m_=$ are the *fuzzy operators* according to the criteria m. The fuzzy operator $\mu^m_=(x, y)$ is computed for any $x, y \in \mathbb{R}$ as follows:

$$\mu^m_=(x, y) = \exp(\frac{\ln \chi}{\epsilon}(x - y)^2)$$

where $\epsilon < 0$ and $0 < \chi < 1$ are parameters that control the degree of fuzziness. The ϵ and χ might be set differently for each criteria $m \in M$. Then, we compute the second fuzzy

operator $\mu^m_<(x, y)$ for the same x, y by the following equation:

$$\mu^m_<(x,y) = \begin{cases} 1 - \mu^m_=(x,y) & \text{if } (x-y) < 0 \\ 0 & \text{otherwise} \end{cases}$$

To fulfill the condition $\mu^m_=(x, y) + \mu^m_<(x, y) + \mu^m_>(x, y) = 1$ which is required for consistency. We also define a third fuzzy operator $\mu^m_>(x, y)$ as follows:

$$\mu^m_>(x,y) = \begin{cases} 1 - \mu^m_=(x,y) & \text{if } (y-x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Nevertheless, we do not need $\mu^m_>(x, y)$ to compute the degree of dominance $d(j_1, j_2)$. Note that when the degree of domination $d(j_1, j_2)$ is equal to 1, we say that the journey j_1 completely dominates the j_2. Finally, we define the *score function* $s : J \mapsto [0, 1]$ that assigns a score $s(j)$ to a journey j in Pareto set J as follows:

$$s(j) = 1 - \max_{j_i \in J \setminus \{j\}} d(j_i, j)$$

The score $s(j)$ is determined by a journey $j_i \in J \setminus \{j\}$ that dominates the journey $j \in J$ the most. Since we find the computation not trivial, we recommend the original source [42], where the Fuzzy dominance is described in detail.

In our solution of ITNAP-PC, we use the Fuzzy dominance scoring algorithm after obtaining the Pareto set from the McRAPTOR algorithm. We assign a score to every journey in the Pareto set, and then we select the journey with the highest score. This post-processing phase is repeated for every origin-destination pair. Thus, we obtain at most one *selected journey* for each origin-destination pair. In other words, the journey selected from the Pareto set is the same journey that the passenger chooses to travel from the desired origin to the destination. When the Pareto set is the empty set, we consider the origin-destination pair as not feasible.

4.5 Intermodal CRP-McRAPTOR Algorithm

If we combine the CRP and the McRAPTOR, we will obtain the algorithm that finds the Pareto set of journeys in the intermodal transport network for an origin-destination pair. Furthermore, we select the journey for each origin-destination by using the fuzzy dominance scoring algorithm. Also, we add the algorithm that computes KPIs at the end of the combined algorithm. The resulting algorithm we provide is called the *Intermodal CRP-McRAPTOR* algorithm, and it solves the ITNAP-PC as a whole. The pseudo-code of the Intermodal CRP-McRAPTOR algorithm is in Algorithm 1. The algorithm accepts the parameters of the ITNAP-PC described in Section 3.8, as arguments. In the beginning, the function `extractNonPublicModes` extracts the non-public transport modes $\mathcal{M}_{np}$ from the allowed transport modes $\mathcal{M}_a$. Then, the main loop iterates across all origin-destination pairs. The variable o is the origin location, and the variable d is the destination location from the same origin-destination pair. In the main loop, the set of initial transfers T_{init} and the set of neighboring stops L_n are initialized as empty sets. Then, each non-public transport mode

is iterated in the inner loop. The inner loop starts with the function `findNeighboringStops`, which finds the surrounding stops L_n of the origin location within the defined range for the transport mode m that is being processed. Afterward, the CRP algorithm is called to compute the optimal journeys J_{fm} from the origin location to every neighboring stop from L_n. At the end of the inner loop, the first mile journeys J_{fm} are added into the set of initial transfers T_{init}.

Algorithm 1: Intermodal CRP-McRAPTOR algorithm

Input: allowed modes $\mathcal{M}_a$, preferences of the city $\{E_s, \varphi\}$, set of origin-destination pairs L
Output: set of KPIs K
Data: road graph G_r, timetable Υ, supported modes $\mathcal{M}$

$J^* \leftarrow \{\}$
$\mathcal{M}_{np} \leftarrow$ `extractNonPublicModes` $(\mathcal{M}_a)$
for $\{o,\ d\ \} \in L$ **do**
  $T_{init} \leftarrow \{\}$, $L_n \leftarrow \{\}$
  for $m \in \mathcal{M}_{np}$ **do**
    $L_n \leftarrow$ `findNeighboringStops` $(o,\ \Upsilon,\ m)$
    $J_{fm} \leftarrow$ `crpQuery` $(o,\ L_n \cup \{d\ \},\ G_r,\ m)$
    $T_{init} \leftarrow T_{init} \cup J_{fm}$
  end
  $J \leftarrow$ `mcRaptorQuery` $(L_n,\ d,\ T_{init},\ \Upsilon,\ \{E_s, \varphi\},\ \mathcal{M}_a)$
  $j_s \leftarrow$ `fuzzySelectJourney` (J)
  $J^* \leftarrow J^* \cup \{j_s\ \}$
end
$K \leftarrow$ `computeKpis` (J^*)
return K

After the initial transfers T_{init} are collected for currently processed origin-destination pair, the McRAPTOR is called to find Pareto set of optimal journeys J. The McRAPTOR algorithm extends the initial transfers by public transport trips while all specified criteria are optimized. Thus, the Pareto set J contains intermodal journeys. From the Pareto set J, we select only one journey j_s and add it to the set of selected journeys J^*. The journey j_s is the best-ranked journey according to the Fuzzy dominance scoring algorithm. When the main loop is exited, we have a set of selected journeys J^* where at most one journey is present for each origin-destination pair. Finally, the set of KPIs K is computed using the function `computeKpis` which meets the conditions defined in Section 3.7. The set of KPIs K is the result of the Intermodal CRP-McRAPTOR, and it complies with the expected solution of the ITNAP-PC.

In short, the inner loop collects initial transfers T_{init} that are needed for the McRAPTOR algorithm. The main loop collects selected journeys J^* that are required for the computation of KPIs. In the end, the set of KPIs K is computed and returned.

Chapter 5

Implementation

In this chapter, we describe the implementation of the Intermodal CRP-McRAPTOR algorithm and other related components. Firstly, we introduce the format of input data used to model the Road Graph and how we utilize the Open Source Routing Machine. Then, we describe the data format for the Timetable structure. As next, we describe how we implement the Intermodal McRAPTOR algorithm in Java. At the end, we describe the parallel computation that is used to process the travel demand faster.

5.1 OpenStreetMap Data

Since the Road graph is a part of the intermodal transport network, we need data that contain a description of roads in the desired city. Preferably such data that carries information about GPS coordinates of every route segment. The OpenStreetMap[1] (OSM) project provides precisely this type of data for the whole planet Earth. OSM project is a community of voluntary contributors that enter and edit the geographical data into the OSM database. Every building, road, point of interest, and more map data are included in the OSM database, and they are provided under the Open Database license. OSM data of particular regions are available in various formats. We prefer files with the *Protocolbuffer Binary Format* (*PBF*) since the size is smaller, and the processing of the is faster than other available formats. Now, we briefly look at the structure of OSM data; it is a composition of three basic elements:

- **Nodes:** A node is a single point determined by *nodeId* and GPS coordinates (*latitude* and *longitude*). More nodes together can describe the shape of a way or the single location on the map.
- **Ways:** A way is a list of nodes that represent a road, a river, or an area boundary. The way can be open or closed, depending on if it describes a line or an area.
- **Relations:** A relation is a group of elements that describes a relationship between them. For example, a relation can describe a type of route, e.g., a bus route, or a cycle route.

[1] https://openstreetmap.org/

Each element may contain a *tag* which is a key-value pair. Tags describe the meaning of a particular element, which can be a type of building, type of road, etc. In this thesis, we do not need a deep understanding of OSM data, as many tools that process the OSM data already exist. The first example is the tool *Osmconvert*[2], which can convert OSM files between formats or extract OSM data within a specified region. This tool is really useful, since on the Geofabrik's website[3] the smallest region is usually a country. We only need data that are from the area of the city. Thus, we download a PBF file with OSM data for country and use the Osmconvert tool to extract data within the desired rectangle.

Example command to extract OSM data within the Prague area:

```
$ ./osmconvert czech-republic-latest.osm.pbf -o=prague-latest.osm.pbf
-b=14.240707,49.946334,14.709360,50.180942
```

After the command is processed, we still have data containing all elements in the area. Thus, we need to extract only the elements that describes roads. We use another command-line tool called Osmfilter[4] that filters data by specifying OSM tags. Note that Osmfilter does not accept PBF format, but *OSM* or *O5M* formats, hence the *PBF* file should be converted using Osmconvert first.

Example usage of Osmfilter to filter roads:

```
$ ./osmfilter prague-latest.o5m -keep="highway=" -drop="access=no"
-o=prague-roads.o5m
```

In this thesis, we need not to filter the road data by ourselves. We use the Open Source Routing Machine (OSRM) that is accompanied by the tool Osrm-extract that extracts roads from *PBF* file. The detailed description of the usage of the OSRM is Section 5.2.

5.2 Open Source Routing Machine

The Open Source Routing Machine[5] (OSRM) is the open-source engine that searches shortest journeys. The OSRM is designed to work with OSM data and is able to search optimal journeys for non-public transport modes, bicycle, car, and walk. A car mode in OSRM is sufficient for our taxi service transport mode, which we want to include in our analysis. According to OSRM website, there are two algorithms implemented for journey planning, the Contraction Hierarchies (CH) and Multi-Level Dijkstra (MLD). Since the Multi-Level Dijkstra is a name for a query phase in Customizable Route Planning (CRP) [22], we consider the MLD as another name for CRP. As we already described in Chapter 4, CRP and CH have a pre-processing phase. Also, the OSRM is accompanied by command-line tools that pre-process the OSM data. Since the OSRM meets our requirements; we decided to use this engine to compute journeys in the Road Graph. There are two possibilities how

[2] https://wiki.openstreetmap.org/wiki/Osmconvert
[3] https://download.geofabrik.de/
[4] https://wiki.openstreetmap.org/wiki/Osmfilter
[5] http://project-osrm.org/

to integrate the OSRM to another application. One possibility is to use the C++ library, which is more difficult because our application is written in Java. Instead of using the Java Native Interface[6] to call the C++ library, we implement a connector to an HTTP API. There is one consequence of choosing this option. We need a running instance of OSRM while calling the API from our application. The full list of accessible services of HTTP API is the following:

- **Nearest:** A service that returns k-nearest OSM nodes with their GPS coordinates.
- **Route:** A service that finds the fastest journey between GPS coordinates in the specified order.
- **Table:** A service that computes the duration or distance of the fastest journey between all pairs of specified GPS coordinates.
- **Match:** A service that matches noisy GPS traces to the GPS locations of the road network in the most acceptable way.
- **Trip:** A service that solves the Traveling Salesman Problem.
- **Tile:** A service that returns Mapbox Vector Tiles[7].

All we need is to compute the duration of the fastest journey from an origin location to surrounding stops. Thus, we implemented the OSRM connector in our Java application (package: `osrm`) that access the Table service via HTTP requests. As a next step, we need to run an OSRM instance that accepts the HTTP requests. To run the OSRM instance, we need to pre-process the OSM data first. The pre-processing phase differs in accordance to which algorithm (CH or CRP) is used in the query phase. We provide an example of commands which needs to be processed to do the pre-processing phase. In the example, we use the PBF file with OSM data as the input. Also we specify the OSRM profile `foot.lua` that means walk transport mode in our notation.

Example of the pre-processing phase for CH:

```
$ ./osrm-extract prague.osm.pbf -p profiles/foot.lua
$ ./osrm-contract prague.osrm
```

Example of the pre-processing phase for MLD:

```
$ ./osrm-extract.exe prague.osm.pbf -p foot.lua
$ ./osrm-partition.exe prague.osrm
$ ./osrm-customize.exe prague.osrm
```

Then, the algorithm which is used during the computations is specified while running the OSRM instance by the parameter *algorithm*. The instance of OSRM computes journeys for only one transport mode, but we need to compute journeys for more transport modes

[6] https://docs.oracle.com/javase/8/docs/technotes/guides/jni/
[7] https://docs.mapbox.com/vector-tiles/reference/

at once. The solution is to run one OSRM instance for each desired transport mode on a different port.

Example of running the OSRM on a local machine:

```
$ ./osrm-routed.exe -algorithm=CH -ip=127.0.0.1 -port=5000 prague.osrm
```

To ease the process, we created a GitHub repository[8] with a Dockerfile[9]. The Dockerfile assembles a Docker image[10] with the OSRM HTTP API for all profiles within the region of the city of Prague. During the build, the pre-processing phase of OSM data for CH is done. The Docker image is publicly accessible on the DockerHub[11].

The recapitulation of this section is that we implement a HTTP connector that communicates with OSRM API. By using this implementation, we are able to compute optimal journeys in the Road graph. Thus, we implemented the first mile phase of the Intermodal CRP-McRAPTOR. Note that the first mile phase collects initial transfers by searching the optimal journeys for non-public transport modes.

5.3 Timetable Data

To model a public transport network which is the part of the intermodal transport network, we need data to create the Timetable structure. More precisely, we need data that include information about scheduled trips in the public transport network. Also, such data have to include GPS coordinates of stops and roads wherever the public transport vehicles move. For this purpose, Google has introduced the General Transit Feed Specification[12] (GTFS). The GTFS defines a common format for public transport data, and it includes everything we need for modeling the Timetable structure. The data in the GTFS format is a group of text files collected in a ZIP file. The most known public database of GTFS data is accessible via OpenMobilityData's website[13]. Each file in the ZIP describes a particular aspect of the public transport network. The following list is the group of required files of the GTFS format:

- **`agency.txt`:** A list of *institutions* that manage public transport.
- **`stops.txt`:** A list of *stops* in the public transport network where a vehicle picks up or drops off passengers. Each stop has a geographical location associated with the name of the stop and additional properties.
- **`routes.txt`:** A list of *routes* in the public transport network. Each route is determined by *routeId*, name, and type. For example, a route is a bus link, on which several trips are arranged during the day.

[8] https://github.com/fiserto1/osrm-java8-docker
[9] https://docs.docker.com/engine/reference/builder/
[10] https://docs.docker.com/engine/reference/commandline/images/
[11] https://hub.docker.com/r/fiserto1/osrm-backend-openjdk8
[12] https://developers.google.com/transit/gtfs/
[13] https://openmobilitydata.org/feeds

- **`trips.txt`:** A list of *trips* in the public transport network. A trip describes one journey managed by one vehicle on a route. A trip is determined by *routeId*, *tripId*, and optionally a geospatial shape of the journey.
- **`stop_times.txt`:** A list of entries that include arrival and departure times for each stop and vehicle that visits the stop during a day.
- **`calendar.txt`, `calendar_dates.txt`:** Files that determine whether a trip is realized on a certain day of a week. Either one of these files is required in the GTFS dataset.

Unless required files, there is another file `transfers.txt` that is usually included in the GTFS dataset. It is a list of *transfers*, where the transfer is the relation between two routes, where it is possible to switch trips. In our implementation, we add the duration of the transfer to the total duration of a journey.

To load the GTFS files, we used the existing implementation that is available in the GitLab repository[14]. The implementation is provided by the Artificial Intelligence Center[15] (AIC). We extended this implementation by the class `GtfsRouteType`, which is the enumeration of route types specified by GTFS reference. In our code, all classes that are needed to load the GTFS into Java objects are in the package `cz.tfiser.gtfs`. Then, we used the same GitLab repository to model the Timetable structure. We slightly edited the code and added a class `MultiTransfer`, which models the initial transfers of non-public transport modes, as stated in Algorithm 1 in Section 4.5. The resulting classes are in package `cz.tfiser.raptor.structures.timetable`.

5.4 McRAPTOR

In the Gitlab repository mentioned in the previous section, there is the implementation of the RAPTOR algorithm, which is a single-criteria version of the McRAPTOR algorithm. While the principle is similar, the implementation of the multi-criteria version is more challenging. Thus, we implemented the McRAPTOR algorithm ourselves, except that the code from the repository is used to load the Timetable structure. Our implementation supports optimization of at most four criteria. The criteria journey duration and number of interchanges are optimized by default. In addition, the optimization of occupancy and costs can be enabled. The implementation is in the class `McRaptor` which is located in the package `cz.tfiser.raptor`.

The McRAPTOR is the essential part of the Intermodal CRP-McRAPTOR algorithm, which follows up on the first mile computation from Section 5.2 by finding journeys in the public transport network.

5.5 Intermodal CRP-McRAPTOR

The implementation of the Intermodal CRP-McRAPTOR from Section 4.5 is a composition of components where each component solves a particular part of the problem. The list of general components is the following:

[14] https://gitlab.fel.cvut.cz/kasnezde/raptor
[15] http://aic.fel.cvut.cz/

- **Journey Planner:** The core component that processes the travel demand and returns the selected journey for every origin-destination pair. It includes the McRAPTOR algorithm and the OSRM connector that calls the OSRM API component. Also, it includes the Fuzzy selector that selects the best journey from each Pareto set of optimal intermodal journeys.

- **OSRM API:** A standalone third-party application that provides the OSRM HTTP API for computing a duration of fastest journeys in the Road Graph. Every particular non-public transport mode has its running instance.

- **KPI Computer:** A component that computes key performance indicators of the intermodal transport network. It takes the output of the Journey Planner component and exports the results into files.

Now, we describe the complete process of one analysis instance by using the schema in Figure 5.1. At first, we prepare input data for the Journey Planner. The Travel Demand is the GeoJSON[16] file that contains origin-destination pairs and departure times. The Occupancy Data is the CSV file that defines the level of occupancy for every trip segment. The GTFS data is a group of text files described in Section 5.3. All three inputs are consumed by the Journey Planner component. As next, we prepare data for OSRM. We process the OSM data for each desired OSRM profile (driving, walking, or cycling) as it is described in Section 5.2. Then, we run the instance of OSRM application from the command-line for every OSRM profile. After that, we can run the Journey Planner component by using the class `RequestProcessorMain`. When the Journey Planner finishes the processing, we use the output as the input for the KPI Computer component.

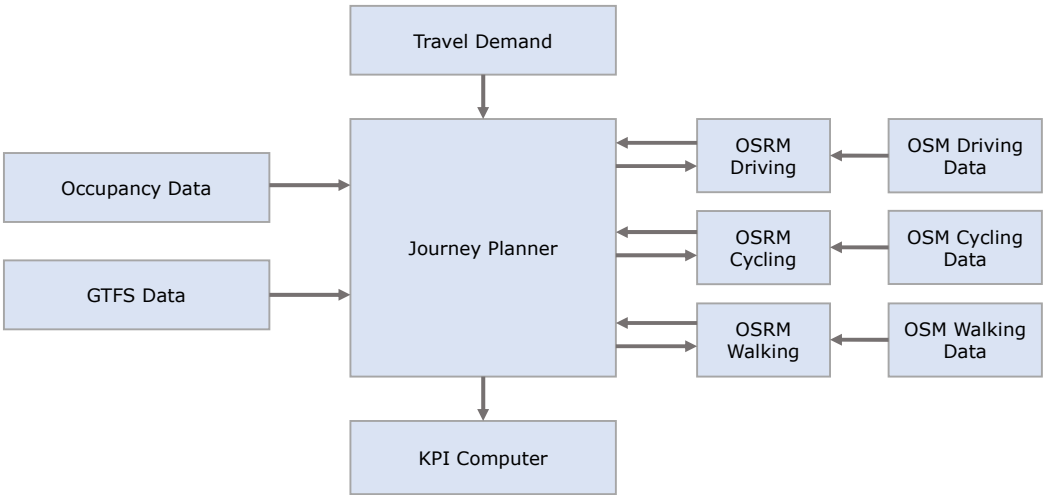


Figure 5.1: The general schema of the implementation.

[16] http://geojson.org/

When we compare our implementation with the pseudo-code in Section 4.5, the inner loop is implemented in the class `FirstMileSolver`, and the main loop is in the class `RequestProcessor`. In the `FirstMileSolver`, the OSRM connector sends queries to OSRM HTTP API. The `RequestProcessor` calls the class `McRaptor`, which computes the Pareto sets. Our implementation of the Fuzzy dominance scoring algorithm located in the class `FuzzyDominancePassengerProfile`. The last part of the pseudo-code is KPI computation; it is located in the class `Analyser`. Our implementation is accessible on GitLab repository[17].

5.6 Parallel Processing

Since the Journey Planner component optimizes four criteria during the computation, it is very demanding on the computation time. We decided to divide the passenger demand into equally large parts. Then, each part is processed on the standalone Journey Planner instance. Although, all instances of Journey Planner use the same OSRM HTTP API. Thus, we still run at most three instances of OSRM application. After all parts of the passenger demand are processed, we use the `ResultMerger` to merge the results of a particular Journey Planner into one file. The schema in Figure 5.2 depicts the parallel processing. In the end, the KPI computer can be used to finish the process.

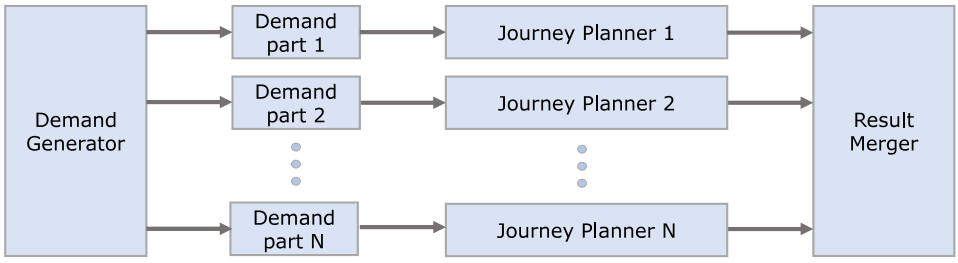


Figure 5.2: The schema of parallel processing of passenger demand.

[17] https://gitlab.fel.cvut.cz/fiserto1/analyser-of-intermodal-transport-network

Chapter 6

Evaluation

In this chapter, we provide an analysis of a real-world intermodal transport network. Firstly, we describe data from which the intermodal transport network is modeled. Then, we determine values for the parameters that are needed to run our application. Finally, we present the outcome of the analysis itself. The analysis is divided into four experiments. In the first experiment, we collect the occupancy data which are needed to detect the inconvenient trip segments. In the following experiments we process the same travel demand while changing the set of allowed transport modes in the intermodal transport network. To process the travel demand, we use the Journey Planner application from Section 5.5. By using the parallel processing (see Section 5.6), we run all the experiments on a remote computing machine[1] provided by Metacentrum[2]. The hardware specification that we used to request a machine in the grid is: 12 CPU and 64 GB of RAM.

6.1 Input Data

We have chosen the city of Prague as an area for our analysis. This area is sufficiently large and contains many transport modes. In this evaluation, we explicitly include these transport modes: bus, tram, subway, ferry, taxi service, bicycle, and walk. For all experiments, we set the departure time window from 7 a.m. to 8 a.m. which simulates the peak hour in the morning.

6.1.1 Transport Modes

In the analysis, we want to show the difference of key performance indicators when we change the set of allowed transport modes in the intermodal transport network. Thus, we experiment with different settings of the set of allowed transport modes $\mathcal{M}_a$, which is defined in Section 3.8.

[1] http://metavo.metacentrum.cz/pbsmon2/hardware#alfrid-cluster.meta.zcu.cz
[2] http://metacentrum.cz

6.1.2 GTFS Data

The General Transit Feed Specification (GTFS) is a data format for public transport timetables. The GTFS data are composed of text files that contain detailed description of public transport entities (e.g., stops, routes, and trips) and their precise GPS locations. We have used the GTFS data provided by Pražská integrovaná doprava.[3] From this dataset, we have extracted only data that are inside the bounding box shown in Figure 6.1. Obtained data are used to represent the public transport part of the intermodal transport network. The resulting numbers of entities are in Table 6.1. We also provide a view of GTFS stops categorized by the transport mode in Figure 6.2.

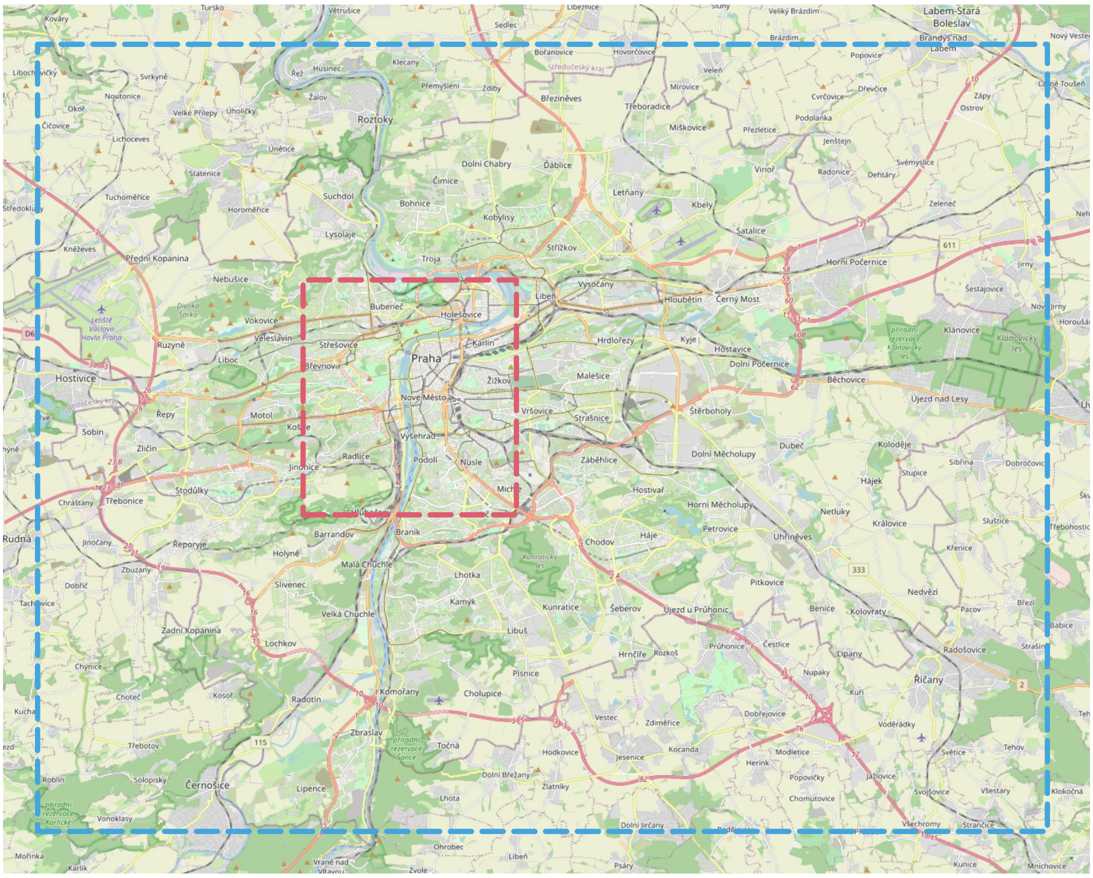


Figure 6.1: The outer bounding box of Prague that is used for data extraction (blue). The outer bounding box has approximately 34 km in width and 26 km in length. The red bounding box describes the destination area for commuters.

[3]https://pid.cz/o-systemu/opendata/

# Stops	# Routes	# Trips	Time Period
4,755	375	72,109	21/02/2020 - 01/03/2020

Table 6.1: The description of GTFS data obtained Pražská integrovaná doprava after extracting the data in the outer bounding box of the Prague area.

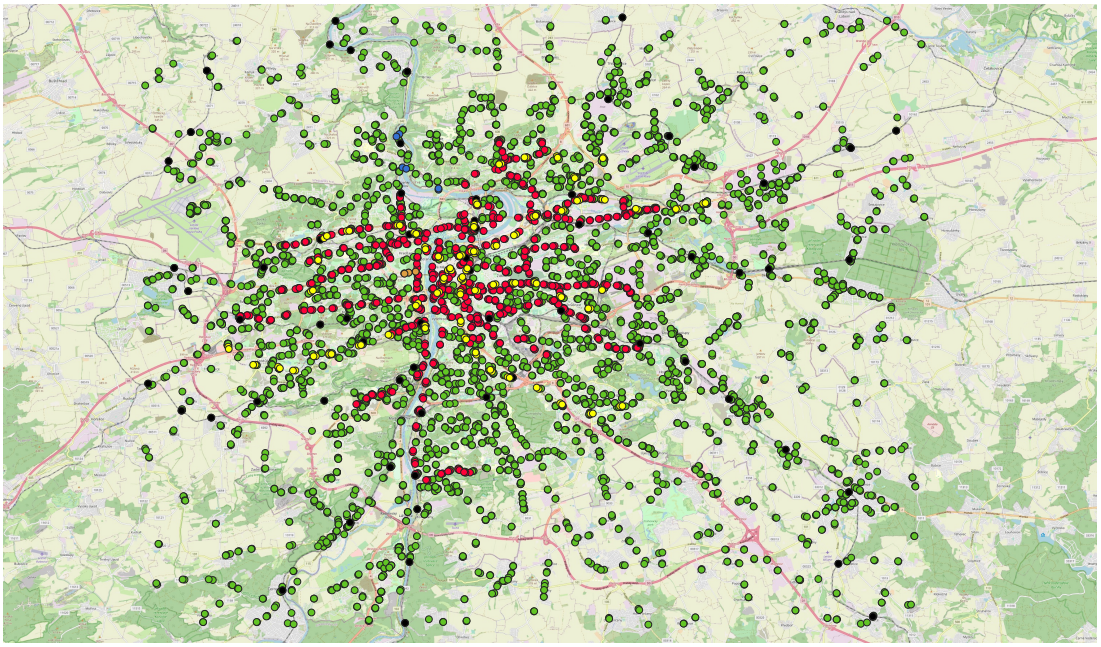


Figure 6.2: The GTFS stops in the outer bounding box. The stops are categorized by the transport mode. (Green - Bus, Red - Tram, Yellow - Subway, Black - Rail, Blue - Ferry, Brown - Cable car)

6.1.3 OSM Data

As the source with OpenStreetMap (OSM) data, we use the website that is managed by Geofabrik[4]. Although there are no data for the Prague area but for the Czech Republic as a whole. From the dataset for the Czech Republic, we extract data within the bounding box as we did with GTFS data. Then, we use features accompanied by Open Source Routing Machine (OSRM) to extract routes and compute route contractions. After that, we pass the data as a parameter to run the OSRM. These data form the Road graph which is the part of the intermodal transport network.

6.1.4 Demand Data

Since we do not know the real travel demand in the morning peak hour, we generate origin-destination pairs randomly. Each origin location is inside the outer (blue) bounding box

[4]https://download.geofabrik.de/europe/czech-republic.html

but outside the inner (red) bounding box in Figure 6.1. On the other hand, all destination locations are in the inner (red) bounding box. As the time of departure, we randomly assigned a time to each origin-destination pair from the time window from 7 a.m. to 8 a.m. In all experiments, we use the same travel demand with 160 000 origin-destination pairs.

6.1.5 Occupancy Data

As we discussed in Section 3.5, preference of the city is to avoid the overcrowding of the public transport network. To consider this in the analysis, we need data that determines the level of occupancy to each trip segment. We were not able to find a source with these type of data. Therefore, we generate a dataset by computing optimal journeys for origin-destination pairs and by collecting occupancy data of each trip segment.

6.2 Journey Planner Configuration

Parameter	**Value**	**Description**
$\mathcal{R}_{max}$	4 rounds	Number of rounds for the McRAPTOR algorithm i.e., returned journey has at most 3 transfers between PT modes and one transfer from NPT mode to PT mode.
$\mathcal{D}^{ts}$	8 km	Maximal distance reachable by taxi from the origin location.
$\mathcal{D}^{bc}$	4 km	Maximal distance reachable by bicycle from the origin location.
$\mathcal{D}^{w}$	2 km	Maximal distance reachable by walk from the origin location.
$\overline{\mathcal{S}}^{t}$	35 km/h	Average speed of a taxi in a city.
$\mathcal{P}^{pt}$	3 €	Price of a single ticket that is required to buy before boarding to the first PT vehicle i.e., a transfer with the same ticket is allowed.
$\overline{\mathcal{P}}^{t}_{km}$	1 €/km	Average price per kilometer for a taxi in a city.
b_{conv}	67 % of max. capacity	Maximum convenient occupancy (see Section 3.7.5). The parameter is defined relatively to a maximum capacity of particular transport mode.
c^{s}	1,500 pers.	Maximum capacity of the *subway* transport mode.
c^{b}	100 pers.	Maximum capacity of the *bus* transport mode.
c^{t}	170 pers.	Maximum capacity of the *tram* transport mode.
c^{r}	2,000 pers.	Maximum capacity of the *rail* transport mode.
c^{f}	50 pers.	Maximum capacity of the *ferry* transport mode.

Table 6.2: Configuration of the Journey Planner.

To run the analysis, we need to determine values of the input parameters. The parameters that need to be set up for the Journey Planner are listed in Table 6.2, and parameters for the setting of Fuzzy dominance are in Table 6.3. To set up the Intermodal CRP-McRAPTOR algorithm, we need to specify the number of rounds and parameters to compute the passenger

costs for using a taxi service. Then, we need maximal reachable distance for every NPT mode to compute first mile journeys. We also need maximum capacities of particular PT modes to compute the KPIs. In the end, we set up the parameters that are used during the journey selection.

Criterion	χ	ϵ
Journey duration	0.8	60
Journey costs	0.8	2
Number of interchanges	0.1	1
Occupancy	0.8	300

Table 6.3: Configuration of Fuzzy dominance (see Section 4.4)

6.3 Analysis

In this section, we describe four experiments by which we analyze the intermodal transport network. In the first experiment we analyze the network without knowledge of occupancy data while only PT and W modes are allowed. By using the results of the first experiment, we generate occupancy data that are later used in other experiments as the input. In the second experiment, we enable the occupancy criterion and analyze the intermodal transport network with the same transport modes (PT and W). In the third experiment we allow the bicycle transport mode in addition to PT and W modes. In the last experiment, the taxi service transport mode is allowed instead of the bicycle mode. Denote that travel demand is the same for every experiment. In the following text, we use the terms, convenient trip segment, inconvenient trip segment and overcrowded trip segment as it is defined in Section 3.7.5.

6.3.1 Experiment 1: Occupancy Generation

The purpose of this experiment is to analyze the default state of the public transport network and generate the occupancy data for the following experiments. At first, we allow only public transport and walk modes in the intermodal transport network model. Since we do not have any occupancy data yet, the criterion that optimizes occupancy is disabled. Then, we use the Intermodal CRP-McRAPTOR algorithm to process the travel demand and compute KPIs. To detect the inconvenient segments and overcrowded segments, we use parameters that determine capacities for each public transport mode and the maximum convenient occupancy from Table 6.2. After 37 minutes of computation, the Journey Planner has found journey for 157,684 passengers out of a total of 160,000. Journeys for other passengers cannot be found for two reasons. At first, we allow at most four transfers during a journey, thus the passenger requested a journey that may be realized only with five or more transfers. At second, the walking range from the origin location is limited to two kilometers. Thus, a passenger requested a journey from a location that is too far from the public transport network. The average duration traveled by one commuter is 63.8 minutes. It is not surprising value according to the size of the outer bounding box in Figure 6.1. Note that the walking time to reach the first stop and the transfer time are included in the duration of the journey.

As the next step, we divide 157,684 selected journeys into trip segments. Then, we obtain the occupancy of every trip segment by calculating the occurrences of particular trip segment. As a result, we get 59,649 unique trip segments that at least one passenger used to travel. To see how occupied, the segments are, we create a histogram of occupancy, where only inconvenient trip segments are included. The histogram is shown in Figure 6.3.

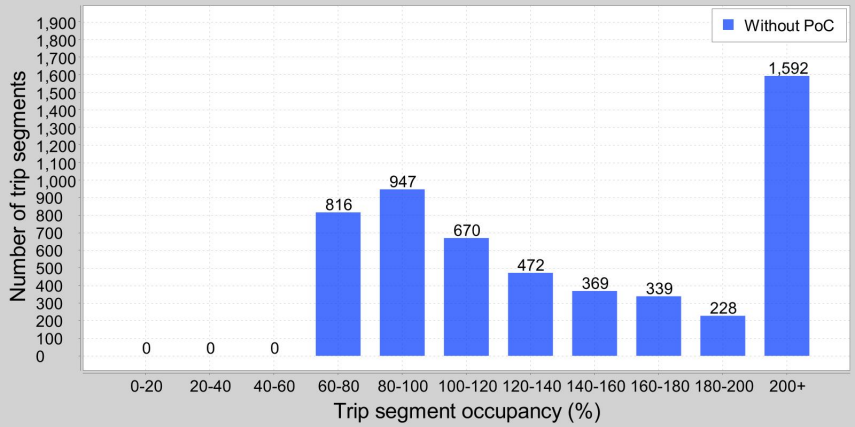


Figure 6.3: Histogram with inconvenient trip segments of all **PT** modes. The intervals of the segment occupancy is on the x-axis. The occupancy is measured as percents from the maximum capacity of the particular public transport mode. The number of segments that belongs to each occupancy interval is on the y-axis

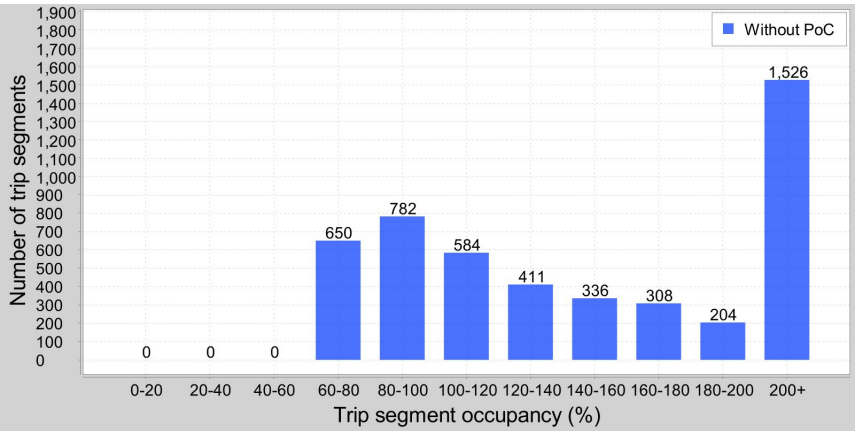


Figure 6.4: Histogram with inconvenient trip segments of the **bus** transport mode. The intervals of the segment occupancy is on the x-axis. The occupancy is measured as percents from the maximum capacity of the particular public transport mode. The number of segments that belongs to each occupancy interval is on the y-axis

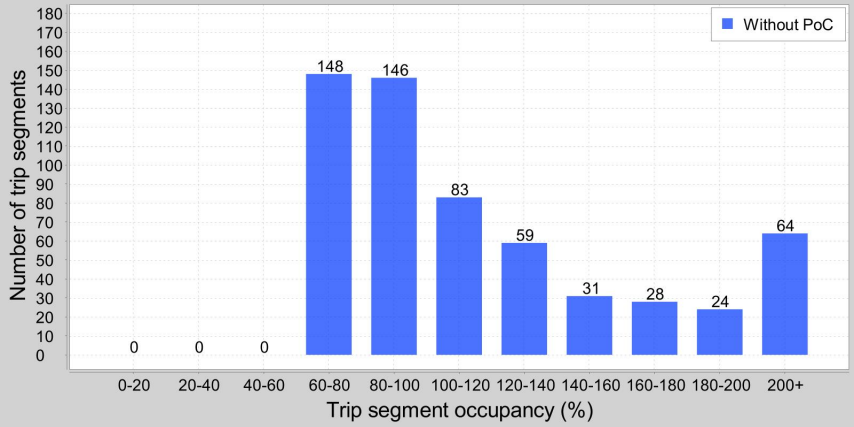


Figure 6.5: Histogram with inconvenient trip segments of the **tram** transport mode. The intervals of the segment occupancy is on the x-axis. The occupancy is measured as percents from the maximum capacity of the particular public transport mode. The number of segments that belongs to each occupancy interval is on the y-axis

In Figure 6.3, we see that 1,592 trip segments are occupied more than twice the maximum capacity of the vehicle; it is 29.3 % of all inconvenient trip segments (5,433). The most of all inconvenient segments belongs to the bus transport mode followed by the tram transport mode. To clarify this, we also show the histograms of the same type for the bus transport mode (see Figure 6.4) and for the tram transport mode (see Figure 6.5).

To summarize the first experiment, we observe that the usage of randomly generated demand lead to 5,433 inconvenient trip segments in the public transport network, and from 5,433 inconvenient trip segments, 3,636 trip segments are overcrowded. Also, we collected the occupancy data to model the preferences of the city (see Section 3.5).

6.3.2 Experiment 2: PT and Walk with Occupancy Criterion

In the second experiment, we process the same travel demand and keep the PT and W modes allowed. Nevertheless, we use the occupancy data obtained from the first experiment to assign a value to every trip segment in the public transport network. After that, we are able to include the occupancy criterion into a set of optimized criteria. This time, the computation time of the Intermodal CRP-McRAPTOR was 3.3 hours. The increase in computation time is caused by adding the fourth criterion. To be exact, more labels are expanded in the McRAPTOR algorithm since the labels are not eliminated by the dominance.

When we look at the composite histogram in Figure 6.6, we observe the trip segment occupancy that is obtained from both experiments. Although, only the segments that were found inconvenient in the first experiment are included in both datasets. In other words, we see how the occupancy has changed on the trip segments that the city marked as inconvenient. The blue color belongs to the first experiment and the yellow color to the second experiment. In the first experiment, we marked 5,433 segments as inconvenient. In the second experiment, 2,636 of them became convenient, and 2,796 segments are inconvenient even after optimization of the vehicle occupancy. When we extract only trip segments with

tram transport mode (see Figure 6.7), we observe that no segment is occupied more than 140 % of the maximum tram capacity. Also, in the interval 80-100 %, more than a half of trip segments were distributed into intervals on the left. Thus, the occupancy in many tram segments was reduced.

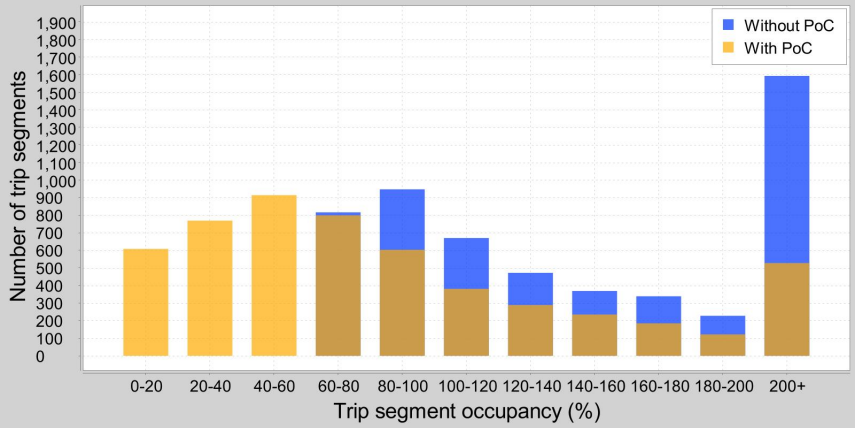


Figure 6.6: Histogram of occupancy in trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **second experiment** are yellow. In the second experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Without PoC). Similarly, the data from the second experiment are after applying PoC (With PoC).

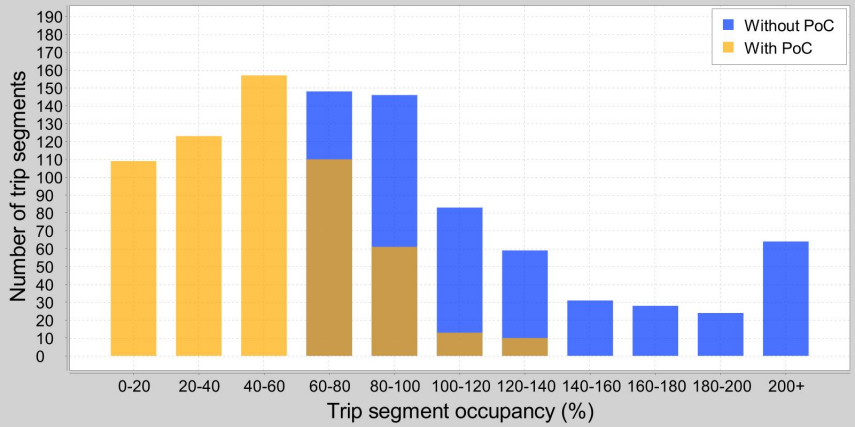


Figure 6.7: Histogram of occupancy in **tram** trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **second experiment** are yellow. In the second experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Without PoC). Similarly, the data from the second experiment are after applying PoC (With PoC).

The KPIs computed for the second experiment are as follows. The algorithm assigned 157,687 journeys in which the average journey duration is 81.6 minutes. In total, 4,507 segments are inconvenient and 2,541 are overcrowded. Thus, when we apply the occupancy criteria, the number of inconvenient and overcrowded segments is decreased. Nevertheless, the average journey duration was increased by 17.8 minutes, which is more then we expected. Note that the optimization distributes occupancy mainly from the trip segments that are marked by city as overcrowded, although it may lead to overload in previously convenient segments. In Figure 6.8, we show all overcrowded trip segments of both experiments on a map of Prague.

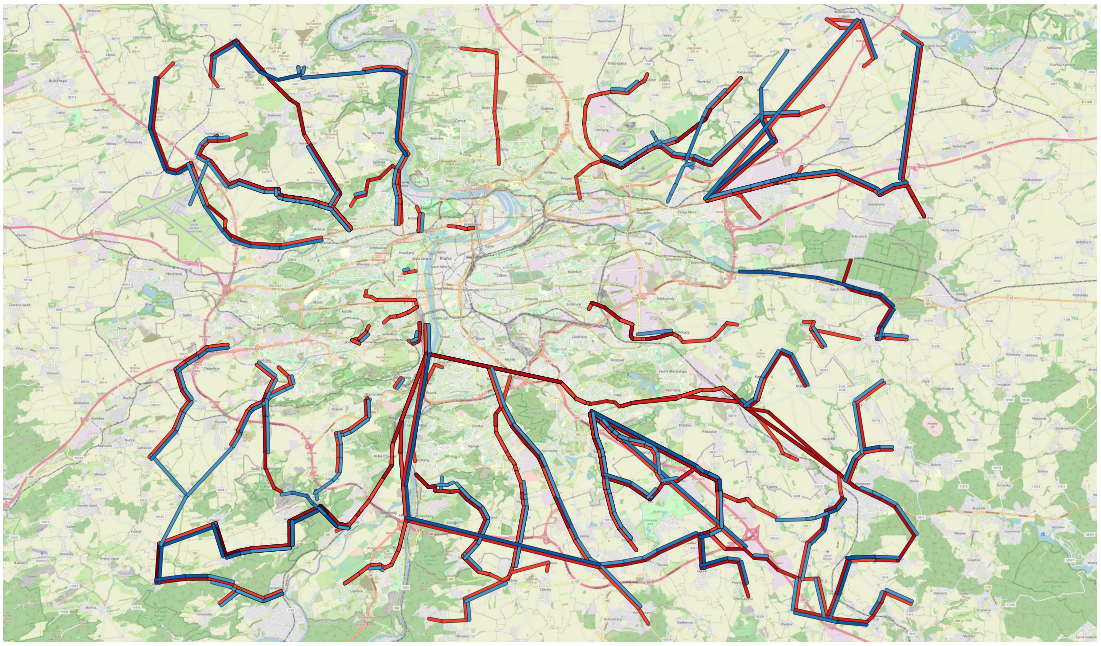


Figure 6.8: Overcrowded trip segments visualized on the map of Prague. The red segments are overcrowded trip segments marked by the city as inconvenient. The blue segments are overcrowded trip segments after the reduction. The segments are categorized from the lowest (light color) to the highest (dark color) occupancy.

6.3.3 Experiment 3: PT, Walk and Bicycle with Occupancy Criterion

In the third experiment, we optimize occupancy with the addition of a bicycle transport mode i.e., we allow the bicycle transport mode along with PT and W modes in the intermodal transport network. The algorithm was processing the travel demand almost for 4 hours. Note that the addition of the bicycle transport mode allowed the Intermodal CRP-McRAPTOR algorithm to find journeys for 159,999 origin-destination pairs. The increase of the number of found paths is caused by the fact, that the commuters more than two kilometers away from public transport are able to access to the public transport network with a bicycle.

Similarly to the second experiment, we observe the change in segment occupancy by using histograms. A histogram of the occupancy of inconvenient trips is in Figure 6.9 and histograms of the occupancy of Bus and Tram trip segments are in Figures 6.10 and 6.11, respectively.

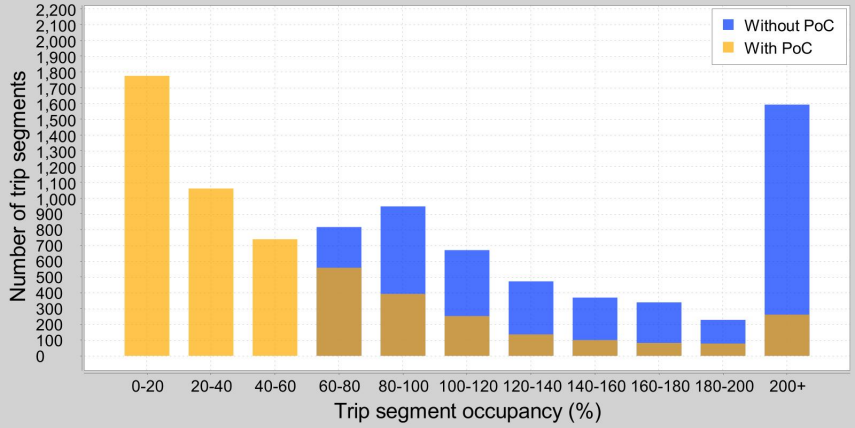


Figure 6.9: Histogram of occupancy in trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **third experiment** are yellow. In the third experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Without PoC). Similarly, the data from the third experiment are after applying PoC (With PoC).

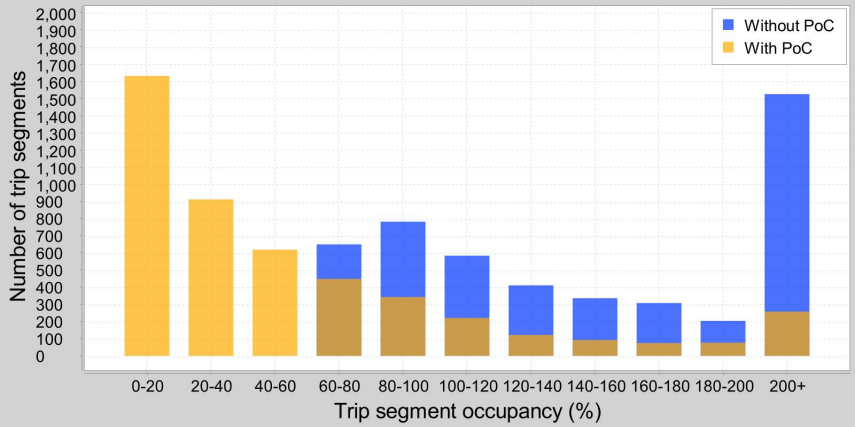


Figure 6.10: Histogram of occupancy in **bus** trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **third experiment** are yellow. In the third experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Without PoC). Similarly, the data from the third experiment are after applying PoC (With PoC).

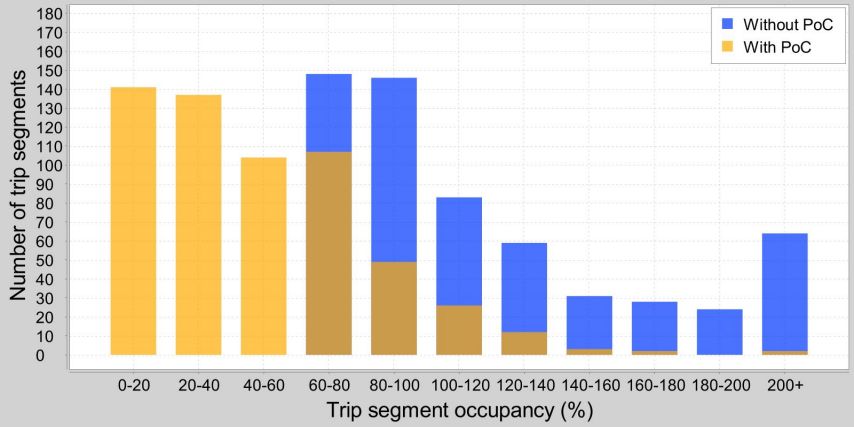


Figure 6.11: Histogram of occupancy in **tram** trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **third experiment** are yellow. In the third experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Without PoC). Similarly, the data from the third experiment are after applying PoC (With PoC).

As well as in the second experiment, we observe that the inconvenient segments are distributed from the right side to the left side of the histogram. The high increase in the interval from 0 to 20 % means, that many inconvenient segments become nearly unoccupied, since the commuters are preferring a bicycle before public transport.

When we observe the total occupancy KPIs measured in the second experiment, 2,818 trip segments are inconvenient and 1,353 are overcrowded. Both KPIs are lower than in the second experiment. Furthermore, the average journey duration (57.5 minutes) is even better than in the first experiment. As we expected, riding a bicycle is an efficient way to ease the overcrowding in the public transport network. The reason for such reduction in average duration of journey, is that some commuters choose a bicycle (instead of walking) to connect to the surrounding stops of public transport.

6.3.4 Experiment 4: PT, Walk and Taxi with Occupancy Criterion

In the fourth experiment, we allow the taxi service, walk and PT transport modes. As we did in the second and the third experiment, the analysis is based on reducing the occupancy in public transport trips. Hence, the occupancy data from the first experiment are used to determine preferences of the city.

The Journey Planner has found journeys for all 160,000 origin-destination pairs after 12 hours of computation. Here, the computation takes much longer then in other experiments. The reason is that using a taxi service is much faster than other analyzed modes whereas journey costs are higher. Recall that we optimize four criteria, duration, number of interchanges, costs, and occupancy. Thus, when the taxi service mode is included in allowed transport modes with PT and W, the domination of labels is very affected. It means,

that the expanded labels in the McRAPTOR part are rarely dominated. The authors of the McRAPTOR algorithm have already pointed out this issue in [36].

One more time, we compare the occupancy of trip segments by using histograms. In Figure 6.12, we observe that the major part of inconvenient segments from the first experiment has the occupancy between 0 % and 60 % of the maximum vehicle capacity. More precisely, 4,545 of 5,433 trip segments are in the interval from 0 to 60 %. The number of bus segments in the interval 0-20 % shows that many citizens has used the taxi service to avoid the inconvenient bus segments. In comparison to Figure 6.9 in the third experiment, we see that the improvement of overcrowding is noticeable.

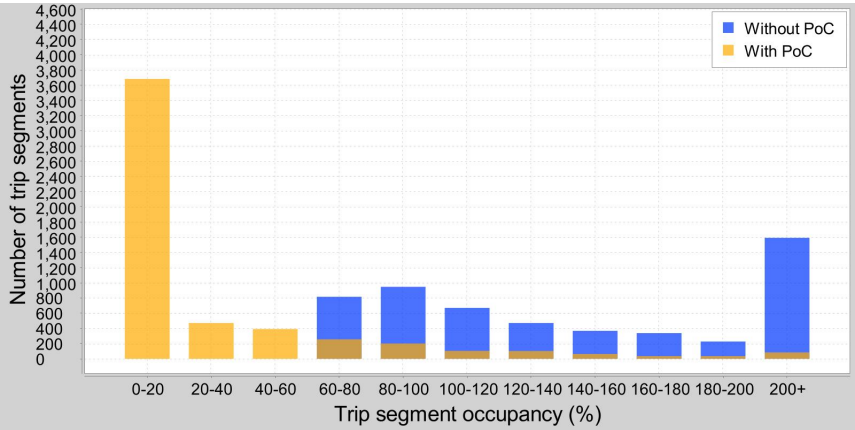


Figure 6.12: Histogram of occupancy in trip segments that are marked inconvenient by the city. The occupancy of trip segments from the first experiment are blue and the occupancy of trip segments from the **fourth experiment** are yellow. In the fourth experiment, the vehicle occupancy is optimized. We say that the occupancy data from the first experiment are before applying the preferences of the city (Before PoC). Similarly, the data from the third experiment are after applying PoC (after PoC).

In the end of this experiment we evaluate the KPIs, the average duration of the journey is 50.8 minutes per passenger and the average journey costs are 3.2 €. As we expected, the average of journey duration is lower, and the average journey costs are higher than in the third experiment where the additional transport mode was bicycle. The total count of inconvenient trip is 1,442, and the number of overcrowded segments is 749 which is the lowest value among all four experiments. Note that the usage of taxi service is probably higher than in the reality. Nevertheless, we would like to regulate the KPIs by changing the input parameters of taxi service in future work as a future work.

6.3.5 Summary

Finally, we provide the summary of KPIs for all four experiments in Table 6.4. In the second column, we observe the values for the first experiment. The 157,684 of origin-destination pairs has feasible journey and detected 5,433 inconvenient trip segments with an average duration of 63.8 minutes per one commuter. In the second experiment where we optimize

vehicle occupancy has processed a similar amount of requests and detected 4,507 inconvenient segments. The decomposition of overcrowding caused that citizens are traveling by less occupied buses. The average duration is 81.6 minutes per commuter, which is circa a 18 minutes prolongation for every commuter on average. In the third experiment, we allowed commuters to use bicycles. It led to finding of journeys for 159,999 origin-destination pairs, which is the whole demand except one origin-destination pair. In this case, 2,818 inconvenient trip segments were detected. The very compelling achievement is that we found more journeys, and the rate of overcrowding in vehicles is still better than in earlier experiments. The foreseen result is that the usage of bicycles improves the average journey duration per commuter to 57.5 minutes. Even better KPIs were computed in the fourth experiment, where a commuter travels 50.8 minutes on average, and 1,422 trip segments are inconvenient. It is almost two times lower occurrences of inconvenient segments than in the third experiment. Nevertheless, introducing the taxi service in the fourth experiment led to the increase of the average spending for a journey.

Key Performance Indicator	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
k_n: Number of journeys	157,684	157,687	159,999	160,000
k_{dt}: Total duration of journeys (hours)	167,567	214,463	153,263	135,476
k_{ct}: Total costs of journeys (€)	472,518	472,527	475,617	512,640
$\overline{k_d}$: Average journey duration (minutes)	63.8	81.6	57.5	50.8
$\overline{k_c}$: Average journey costs (€)	2.997	2.997	2.973	3.204
$\overline{k_{noi}}$: Average number of interchanges	2.9	2.8	2.5	2.3
k_u: Number of used trip segments	59,649	73,878	56,897	54,537
k_i: Number of inconvenient trip segments	5,433	4,507	2,818	1,442
k_o: Number of overcrowded trip segments	3,636	2,541	1,353	749
k_{su}: Number of overcapacity trip segment usages	553,016	252,133	124,960	84,368

Table 6.4: Summary table of all measured KPIs (see Section 3.7) in all four experiments.

Chapter 7

Conclusion

In this work, we have analyzed the intermodal transport network while considering preferences of the city. To accomplish the goal, we proceeded with the following steps. Firstly, we have researched similar work to understand the problem properly. Then, we provided a representation of the intermodal transport network as a composition of the public transport network and the road network. Within the network representation, we defined the intermodal transport network analysis problem with preferences of the city. As the next step, we described our approach to solve the problem. Last but not least, we implemented all features needed to process the analysis, Demand Generator, Journey Planner, Result Merger, KPI Computer. The Demand Generator randomly generates the origin-destination pairs. The Journey Planner searches journeys for origin-destination pairs while optimizing four criteria, duration, occupancy, costs, and number of interchanges. The KPI Computer computes key performance indicators (KPIs) and outputs results into files. Lastly, we evaluated our approach in the area of Prague city. We have analyzed the scenario where commuters are traveling in peak hour towards the city center. We have compared the KPIs of four experiments with different settings of the intermodal transport network. We have observed that even with imperfect demand, our algorithm ease the occupancy of the public transport network by offering commuters slightly slower journeys in order to avoid overcrowded trips. The rate of occupancy reduction depends on the types of transport modes that are allowed for commuters to travel. In the fourth experiment where the taxi service transport mode is allowed, we achieved the reduction in the number of overcrowded segments by 79.4 %.

7.1 Future Work

This thesis shows the potential to analyze and optimize the overcrowding in public transport. There are plenty of options move this work forward. We would like to generate a realistic travel demand and show the real value of our analyzing tool. Also, we would like to process the analysis with other transport modes, e.g. electric scooters. Another option to go further, is to improve the algorithm in Journey Planner to have better computation time.

7.1.1 Data Gathering

Although it is out of the scope of this thesis, it would be excellent to work with a realistic demand. In future, we would like to aim at reproducing the travel demand that better corresponds to reality. If might be done by gathering the data from traffic measurement reports or a census. If we manage to simulate the real demand it would satisfy our efforts. Similarly important is the gathering of data about other transport modes. With each introduction of a new transport mode, we need data about the demand accompanied by extra data e.g., locations of bike racks or park and rides. We want to do analysis even with modern transport modes including bike-sharing and electric scooters. That would complete our idea of the real intermodal transport network.

7.1.2 Improvements of Journey Planner

Our multi-criteria Journey Planner optimizes four criteria at once, occupancy, duration, number of interchanges, and costs. The Journey Planner supports all public transport modes and above that also walk, bicycle, and taxi service. Except for adding support of other transport modes, we want to improve computation of the criteria that varies for each transport mode, e.g. costs of a journey. The multi-criteria computation is tremendously costly on the processing time of the analysis. Our plan is to include implementation of Restricted Pareto Sets from [40], it should significantly speed up the Journey Planner.

7.1.3 Improvements of Analysis

The analysis in this work is based only on the analysis of the overcrowding in the public transport vehicles, but the utilization of our algorithm is wide. With small changes, we can optimize different criteria of journey and then compute other key performance indicators of the intermodal transport network. In future, we would like to explore more preferences of the city. Then, we may adapt our algorithm and analysis to satisfy these preferences. Also, we would like to automatize process that starts with the data insertion and ends with the graphical outputs.

Bibliography

[1] J. Hrnčíř. Models and algorithms for sustainable journey planning. Doctoral thesis, Czech Technical University in Prague, 2016.

[2] F. Rudolph and N. Szabo. Multimodal analysis methodology of urban road transport network performance. FLOW Project Consortium, 2016.

[3] Transportation Research Board, Engineering National Academies of Sciences, and Medicine. *Multimodal Level of Service Analysis for Urban Streets: Users Guide*. The National Academies Press, Washington, DC, 2008.

[4] J. Nykl. Agent-based public transport network analysis. Bachelor's thesis, Czech Technical University in Prague, 2013.

[5] C. von Ferber, T. Holovatch, Yu. Holovatch, and V. Palchykov. Public transport networks: empirical analysis and modeling. *The European Physical Journal B*, 68(2):261–275, 2009.

[6] J. Gil. *Building a Multimodal Urban Network Model Using OpenStreetMap Data for the Analysis of Sustainable Accessibility*. 2015.

[7] X. Leung, S. Chan, and P. Hui P. Lio. Intra-city urban network and traffic flow analysis from gps mobility trace. 2011.

[8] S. Gao, Y. Wang, Y. Gao, and Y. Liu. Understanding urban traffic-flow characteristics: A rethinking of betweenness centrality. *Environment and Planning B Planning and Design*, 40:135 – 153, 2013.

[9] M. Kurant and P. Thiran. Extraction and analysis of traffic and topologies of transportation networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 74:036114, 2006.

[10] T. Shanmukhappa, I. W. H. Ho, C. Tse, X. Wu, and H. Dong. Multi-layer public transport network analysis. pages 1–5, 2018.

[11] A. London, A. Háznagy, F. Istvan, and T. Nemeth. Complex network analysis of public transportation networks: A comprehensive study. 2015.

[12] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[13] X. Wu, H. Dong, C. K. Tse, I. W. H. Ho, and F. C. M. Lau. Analysis of metro network performance from a complex network perspective. *Physica A: Statistical Mechanics and its Applications*, 492:553 – 563, 2018.

[14] T. Shanmukhappa, I. W. H. Ho, C. Tse, and K. Leung. Recent development in public transport network analysis from the complex network perspective. *IEEE Circuits and Systems Magazine*, 19:4880–4897, 2019.

[15] J. Scheurer, C. Curtis, and S. Porta. Spatial network analysis of public transport systems: Developing a strategic planning tool to assess the congruence of movement and urban structure in australian cities. *30th Australasian Transport Research Forum*, 2007.

[16] J. Scheurer and C. Curtis. Spatial network analysis of multimodal transport systems: Developing a strategic planning tool to assess the congruence of movement and urban structure. Curtin University Sustainability Policy Institute (CUSP) and Australian Centre for Governance and Management of Urban Transport (GAMUT), 2008.

[17] H. Bast, D. Delling, A. V. Goldberg, M. M. Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. *CoRR*, abs/1504.05140, 2015.

[18] J. Dibbelt. Engineering algorithms for route planning in multimodal transportation networks. Dissertation, Karlsruhe Institut of Technology (KIT), 2016.

[19] T. Pajor. Algorithm engineering for realistic journey planning in transportation networks. Dissertation, Karlsruhe Institut of Technology (KIT), 2013.

[20] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, pages 269–271, 1959.

[21] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th International Conference on Experimental Algorithms*, WEA'08, page 319–333, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning. In *Experimental Algorithms*, pages 376–387, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[23] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning in road networks. *Transportation Science*, 51(2):566–591, 2013.

[24] R. Bauer and D. Delling. SHARC: Fast and robust unidirectional routing. *J. Exp. Algorithmics*, 14, 2010.

[25] A. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[26] J. Hamme. Customizable route planning in external memory. Bachelor's thesis, 2013.

[27] E. Q. V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236 – 245, 1984.

[28] C. Dean. Continuous-time dynamic shortest path algorithms. Master's thesis, Massachusetts Institute of Technology, 1999.

[29] B. S. Stewart and C. C. White. Multiobjective A*. *J. ACM*, 38(4):775–814, 1991.

[30] L. Mandow and J. L. P. De La Cruz. Multiobjective A* search with consistent heuristics. *J. ACM*, 57(5), 2008.

[31] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[32] C. T. Tung and K. L. Chew]. A multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62(2):203 – 209, 1992.

[33] E. Machuca and L. Mandow. Multiobjective heuristic search in road maps. *Expert Systems with Applications*, 39(7):6435 – 6445, 2012.

[34] G. S. Brodal and R. Jacob. Time-dependent networks as models to achieve fast exact time-table queries. *Electronic Notes in Theoretical Computer Science*, 92:3 – 15, 2004.

[35] F. Schulz, D. Wagner, and K. Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. In *Algorithm Engineering*, pages 110–123, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[36] T. Pajor D. Delling and R. F. Werneck. Round-based public transit routing. Society for Industrial and Applied Mathematics, 2012.

[37] D. Delling, J. Dibbelt, T. Pajor, and R. F. Werneck. Public transit labeling. In *Experimental Algorithms*, pages 273–285, Cham, 2015. Springer International Publishing.

[38] O. Dib, M. A. Manier, L. Moalic, and A. Caminada. A multimodal transport network model and efficient algorithms for building advanced traveler information systems. *Transportation Research Procedia*, 22:134 – 143, 2017. 19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey.

[39] F. Dalkılıç, Y. Dogan, D. Birant, A. Kut, and R. Yilmaz. A gradual approach for multimodel journey planning: A case study in Izmir, Turkey. *Journal of Advanced Transportation*, 2017:14, 2017.

[40] D. Delling, J. Dibbelt, and T. Pajor. Fast and exact public transit routing with restricted pareto sets. In *ALENEX*, 2019.

[41] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck. Computing multimodal journeys in practice. In *Experimental Algorithms*, pages 260–271, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[42] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck. Computing and evaluating multimodal journeys. Technical report 2012-20, Karlsruhe Institut of Technology, 2012.

[43] D. Kirchler. Efficient routing on multi-modal transportation networks. Master's thesis, Ecole Polytechnique X, 2013.

Appendix A

CD Content

The attached CD contains source files with the implementation of the Intermodal CRP-McRaptor algorithm. The implementation is written in Java (ver. 1.8). We also included the runnable (JAR) file that is located in the directory: `..\target`

To run the application, follow these steps:

1. Copy the directory `..\target` to your directory. (The directory path should not contain any spaces.)
2. Download the OSRM application from: http://build.project-osrm.org/
3. Do the pre-processing phase of the OSRM as it is described in Section 5.1. (Use the PBF file: `..\target\prague.osm.pbf`)
4. Run the OSRM HTTP API for walk transport mode using the command:
   `$ ./osrm-routed.exe -ip=127.0.0.1 -port=5002 prague.osrm`
5. Run the Journey Planner by the following command:
   ```
   $ java -cp analyser-of-itn-1.0.0.jar \
   cz.tfiser.analyser.DemoMain \
   travel-demand/example-requests.geojson &> analyser.log
   ```

In file `example-requests.geojson`, there is an example of travel demand with 100 origin-destination pairs. If this file is passed as and argument of the JAR application, the analysis of intermodal transport network including the PT and W transport modes will be processed. After the completion, output files are stored into the `..\target\prague` directory.

The configuration files for the application are located in `..\target\config`. You can enable or disable occupancy criterion by setting the property in `application.properties`:

- `request-processor.occupancy-criterion.enabled`