FACHHOCHSCHULE KÄRNTEN

MASTER THESIS

Evaluating the effectiveness of real-time information in multimodal public transport trip planning

Submitted in partial fulfilment of the requirements of the degree
Master of Science in Engineering

Spatial Information Management

Carinthia University of Applied Sciences
School of Engineering & IT
Department of Geoinformation & Environmental Technologies

Author: Daniel Steiner, BSc

Supervisor: FH-Prof. Mag. Dr. Gernot Paulus MSc., MAS
School of Engineering & IT, Department of Geoinformation & Environmental Technologies,
Carinthia University of Applied Sciences, Villach, Austria

Supervisor: Prof. Dr. Hartwig Hochmair
Fort Lauderdale Research and Education Center,
Geomatics Program, University of Florida, Davie, Florida

Villach, September 2014

Science Pledge

I hereby declare that

- the Master Thesis has been written by myself without any external unauthorised help and that it has not been submitted to any institution to achieve an academic grading.
- I have not used sources or means without citing them in the text; any thoughts from others or literal quotations are clearly marked.
- the enclosed Master Thesis is the same version as the version evaluated by the supervisors.
- one copy of the Master Thesis is deposited and made available in the CUAS library (§ 8 Austrian Copyright Law [UrhG]).

Villach, 15.09.2014

Daniel Steiner

Acknowledgements

First of all, my appreciation refers to my parents who provided inimitable opportunities for my education, as well as my family and all my friends who supported me during the whole study.

Besides, I am deeply grateful and thankful to my supervisor Prof. Dr. Hartwig Hochmair, Fort Lauderdale Research and Education Center, Geomatics program, University of Florida, Davie, for giving me the chance to do my exchange semester in the United States. His excellent support and encouragement helped me to broaden my skills in scientific research and was very helpful in the preparation of my master thesis. He has a great ability to motivate people and to tackle new research problems from different viewpoints.

I also want to show my gratitude to my supervisor FH-Prof. Dr. Gernot Paulus, MSc, MAS, School of Engineering & IT, Department of Spatial Information Management, Carinthia University of Applied Sciences, Villach, Austria, for his valuable support and the detailed feedback. His constructive comments as well as the data provision were very helpful for the preparation of the thesis.

Additionally, I want to thank Sreten Svetojevic, MSc, for his support in topics concerning databases. As my roommate, he was always motivated to help me with all his knowledge. I owe my deepest gratitude to him!

Finally, I want to offer my regards to the Austrian Marshall Plan Foundation, who played an important role during my research at the Fort Lauderdale Research and Education Center. Without their financial support, my stay in the United States would not have been possible.

Abstract

Modern technologies in the public transportation sector offer a variety of opportunities to advance solutions for every-day use. Since users of public transit systems are more and more relying on online information to plan their trips, improved system performance, efficient capacity utilization, accurate trip planning capabilities and a flexible data infrastructure are important for today's transit agencies to meet these needs. A relative new global standard in the public transportation field, called General Transit Feed Specification (GTFS), defines a common format for transit schedules including associated geographic information. These interoperable "feeds", if shared with the public by public transit agencies, can be used by developers to write applications for public transit users.

The goal of this research is to analyze published real-time information for multimodal public transport trip planning, as well as to check the feasibility to integrate such information into a route planner. To accomplish these tasks, the Open Source software OpenTripPlanner (OTP) will be used, which enables the use of GTFS feeds for the route calculation. While GTFS feeds provide timetable data as well as spatial information about routes and stops, OpenStreetMap (OSM) extracts include walking and bicycle information of the research area. All united layers combined with the schedule form the routing graph.

Due to recent developments, it is also now possible to include real-time delay predictions for trip planning with the OTP in areas, where this information is published by the transit agency. This feature has only recently been added to the OTP, and also transit agencies are just beginning to share this kind of information, which requires advanced data collection infrastructure. Thus, the focus of this thesis is on this novel aspect, whereas the main part deals with assessing the effect of real-time data on route planning results. We will define different scenarios that simulate different types of passengers planning a route. The scenarios differ in what type of information is available to the traveller at the beginning and during the trip. Part of this effort is to analyze the data quality of delay predictions provided by transit agencies as part of their Web service. The quality will be evaluated by comparing timestamps from actually recorded vehicle positions at individual stops (provided as a real-time feed that is published as well) with delay predictions at these stops made in advance.

Keywords: public transportation, multimodal route calculation, trip planning, real-time delay predictions

Table of Contents

List of Abbreviations

ADFC	*Allgemeiner Deutscher Fahrrad Club*
API	*Application Programming Interface*
AVL	*Automatic Vehicle Location*
CEN	*European Committee for Standardization*
CSV	*Comma Separated Value*
CUAS	*Carinthia University of Applied Sciences*
DB	*Deutsche Bahn*
DEM	*Digital Elevation Model*
EE	*Enterprise Edition*
EPTP	*Enhanced Profitable Tour Problem*
ESRI	*Environmental Systems Research Institute*
ETL	*Extract/Transform/Load*
GDI	*Geodata Infrastructure*
GIP	*Graphenintegrationsplattform*
GIS	*Geographic Information System*
GMT	*Greenwich Mean Time*
GPS	*Global Positioning System*
GTFS	*General Transit Feed Specification (former: Google Transit Feed Specification)*
IFPOT	*Identification of Fixed Objects in Public Transport*
ILP	*Integer Linear Program*
IM	*Infrastructure Manager*
ISO	*International Standardization Organization*
ITS	*Intelligent Transport Systems*
JEQL	*Java Extended (Embeddable, ETL, Efficient) Query Language*
NP	*Nondeterministic Polynomial Time*
ÖBB	*Austrian Federal Railways / Österreichische Bundesbahnen*
OD	*Open Data*
ODF	*Open Data Foundation*
OSD	*OpenSource Definition*
OSM	*OpenStreetMap*
OTP	*OpenTripPlanner*
PB	*Protocol Buffer*
POI	*Point of Interest*
REST	*Representational State Transfer*
SIRI	*Service Interface for Realtime Information*
TSP	*Travelling Salesman Problem*
UIC	*International Union of Railways*
US	*United States*
USGS	*United States Geological Survey*
VDV	*Verband Deutscher Verkehrsunternehmen*
VGI	*Volunteered Geographic Information*
XML	*Extended Markup Language*
ZIP	*acronym for “zipper”*

1. Introduction

This research uses an Open Source multimodal route planner that is able to integrate public transport timetable data as well as real-time delay predictions. Multimodal planning refers to planning that considers various modes like walking, cycling, automobile, public transit and the connections among these modes (Litman 2012). Another main part deals with the influence of real-time data in the route planning and the temporal impacts for the user. The inclusion of actual data during the route calculation process may have an influence in the route selection, which leads to temporal alternation as well. These outcomes need to be determined and assessed.

The first section contains a brief overview about the motivation of this research, as well as research challenges and project goals. Furthermore, the methods of solution are described and what results are expected. Finally, the structure of the thesis is explained in a short summary.

1.1 Motivation

One important motivation in this project is the availability of Open Source routing software solutions like the OpenTripPlanner (OTP 2013). Using this tool, a free access to the source code is provided to improve the product as well as adapting the software to personal needs. The Open Source Initiative not only guarantees universal access to the code, but also permits a free redistribution and the transmission of rights that are attached to a program without any restriction (OSD 2013). OTP allows users and third-party applications to combine information about bicycle and transit journeys via a built-in web interface using a public Application Programming Interface (API) (OTP 2013). It is comprehensive and highly functional software allowing the calculation of - for example - a bicycle route and taking into account several route optimization criteria including slope, where other optimization criteria can also be implemented if corresponding attributes for the base data are available.

Open Data (OD) is a form of data that offers the possibility to connect different parts of an information chain together. Because data has many sources, raw data commonly go through different editing, aggregation and analytical stages. The characteristics of Open Data should offer an easy performing task like the discovering, accessing, finding, effectively communicating and sharing of data (ODF 2013). Because of the characteristics mentioned, Open Data are very powerful resources that allow developers to access them in a standardized way without time consuming and very expensive processes. Although it is a relatively unexploited and modern instrument for supplying data to the public, more and more institutions are willing to provide the information in a standardized way. A clear definition of the term “Open Data” is given by the Open Definition (2005):

“A piece of data or content is open if anyone is free to use, reuse, and redistribute it — subject only, at most, to the requirement to attribute and/or share-alike.”

The increased availability of OD is an important base content for developing new and innovative applications and therefore another motivation. OpenStreetMap (OSM) is a good example of how to successfully enable the creation of a virtual world map to the public (OSM 2013).
Nowadays, OD sources also include public transportation data like time table information and even real-time information about the transportation vehicles (GoogleTransitDataFeed 2014). Google was the first organization that developed a worldwide standard for the exchange of public transit data, called “General Transit Feed Specification” (GTFS) (Kaufman & Wagner 2012). An extension to this standard is a data format called “protocol buffer” (PB), that enables the combination of real-time information with this standard (Google Realtime 2014). In Europe, Netherlands is the only country that offers its real-time public transit data to the public and will be used for this research. The use of this data could give an answer to the question if real-time information in a route planning system generates an advantage to the user or not. Out of this research, quality factors about the assessment of real-time services can be determined.

1.2 Goals & research problems

This project is split into two parts: The first, smaller part, illustrated the feasibility to combine public transit data and cyling infrastructure in Carinthia. The second, major part evaluates the effectiveness of real-time information in multi-modal public transport trip planning in Netherlands. The Netherlands was chosen, since this was the only reason where real time transit information was made available online to the public at the time of writing this thesis.
There are different types of online portals for cyclists in the World Wide Web. One type of online tourist portals for cycling activities like “Muensterlandradweg.de” (Muenster 2014), “Bergfex” (Bergfex 2013), “Kaernten Werbung” (Kaernten 2013) or „Radtouren.at“ (Radtouren 2014) offers pre-defined cycle tours among which the user can choose one of them to drive.
More advanced portals allow the user to enter a desired start and end point. Examples are “AnachB.at” (Wien 2013), “Google Maps” (Maps 2013) or “Radroutenplaner NRW” (Route NRW 2014). If public transit data is available, some portals also offer the possibility to plan a cycle tour with consideration of train or bus availability, e.g. “Google Maps”, “AnachB.at” or “Verkehrsauskunft Oesterreich” (VA Oesterreich 2014). There are also some finished OTP deployments that can be found in different cities like in Portland (TriMet 2014), Tampa (OTP Tampa 2014) or Amsterdam (OTP NL 2014), which all allow a combination of bike and transit. The possibility to combine OpenStreetMap bicycle layers together with public transit data enables the force of new innovation for online cycling routing platforms. According to the “TourIS Kärnten” website (TourIS, 2013), the trend of bike tourism is growing more and more. A study from “Allgemeiner

Deutscher Fahrrad-Club" (ADFC) in 2006 shows that more than 44% of all Germans used their bike during their holidays. Following a study from 2013, 66% of all cyclists are using the Internet for searching relevant information before starting a tour (ADFC 2013).

For the calculation of user-tailored routes, we have to consider that different types of bicycles exist: road bikes, touring bicycles, mountain/downhill bikes, city bicycles, e-bikes etc., and most of them are designed for different types of streets. So there is a need to tailor trip planning applications to the need of different cycling user groups.

The application that is going to be implemented has the advantage that it makes use of Open Source tools as well as Open Source data. More explicitly, the new system will use the OpenTripPlanner (OTP) engine linked with OpenStreetMap (OSM) data. In an extended version that is going to be implemented, the cycling network will be combined with multimodal open transit data.

Transit data from Open Data sources represent a new approach to develop online routing applications for cyclists. As described at the beginning of the motivation section, these data can be obtained without any costs and the calculation of a route also contains flexible information. At this point of view, the user gets rid of planning cycling trips choosing from fixed and inflexible routes.

In existing cycling routing applications in Austria, open transit data were never connected to an OTP routing application before, since these data are hardly available to the public in this country. As an example, the "Österreichische Bundesbahnen" (Austrian Federal Railways - ÖBB) offered their public transit time tables in a digital format to Google, but not to the public (DerStandard 2013, Kurier 2013). The only way to get open public transit data in Austria is via "Wiener Linien" or "Linz AG Linien". Both companies offer an API interface for real-time public transport data, where the user can request the desired information via Extended Markup Language (XML) format (Wiener Linien 2014, Linz AG Linien 2014). The limitation is that these data are only available in the cities Vienna and Linz.

The main goal of this research is to build a functioning OTP system including transit data. During the starting phase, the main criteria for setting up such kind of an application as well as the workflow of given data needs to be evaluated. A desired extension of the trip planner includes real-time public transit data, where the route planner continuously gets information about actual delays. Jariyasunant et al (2011) addressed the need for real-time information in transit to increase the system reliability which is defined by the relative number of vehicles arriving on schedule. Hickman & Wilson (1995) already demonstrated how real-time information may be incorporated by the passenger in making a path choice decision. They pinpoint the value of uncertainty of transit services to the passenger, which can affect the trip planning. Because of this fact, it is assumed that additional information can improve the passenger's decisions in travelling. As a result of their research, Hickman & Wilson stated that the reduction of travel times using real-time information is very modest (1-3%) and doubt on the value of these systems. This present research makes use of latest technology to assess the quality and the impact of this dynamic information.

1.3 Method of solution

A state-of-the-art literature review is conducted at the beginning, which includes an evaluation of existing bicycle trip planners and related data sources. Besides, a review of existing round trip algorithms could offer an incentive for more innovation. In a further step, a review of terms like “multimodal routing” needs to be performed to get a sense for how to work with different types of street networks. Multimodal means that the chosen route has to respect characteristics like multiple capacity constraints and multiple time windows (Moccia et al 2008). In the case of commuting with a train, there may be a big difference if one vehicle is arriving too early and the other one arrives too late. It is important to know the structure of transit data and what data is available to work with. Summing up, the concept of multimodal routing represents a solution approach. Another review has to include OSM in terms of bicycle completeness. A study of Hochmair et al (2013) analyzed the completeness of OSM cycling data as well as the completeness of bicycle features in OSM and Google Maps data through data comparison. They found out that the amount of mapped cycling data is likely to grow even more in the near future. It is also stated that European cities have a large network of bicycle trails, while the United States (US) has a higher density of designated lanes.

As related to the collected information from the literature review, a further step includes the conceptual modelling. In order to realize the routing function for cyclists, a data model has to be created to define the types of data being used. The creation of a criteria catalogue supports the decision of what road features and attributes to include. Terms like cycle lanes in one way, cycle lanes in two-ways, cycle tracks or bicycle restrictions are only a few ones out of a huge variety of settings that need to be considered to verify the use of the desired routing functionality (WIKI Bicycle 2013).

Further steps include the analysis of geodata, including what data is available for testing purposes and what data can be used for the implementation of the prototype. The main data source will be OSM, where cycling data is available all over Carinthia and can be used comprehensively. Another data source could be an Open Government Data Portal like that from Carinthia, which is online since October 29th, 2013 (Data KTN 2014). This source offers data from 14 categories, also including bus network, stopping points, a federal traffic network and bike lanes. Besides, it includes a Digital Elevation Model (DEM) all over Carinthia, which could be used to combine the OTP with slope information. Another data provider for DEMs could be the Global Data Explorer (USGS 2013). The final data to be acquired are the transit data, which contain information about the public transportation, all transport plans as well as the stations and networks. The only providers that offer this information as OD in Austria are “Wiener Linien” (Wiener Linien 2014) and “Linz AG Linien (Linz AG Linien 2014). The access to this data is granted after a request via a web API to the operator.

All data sources like OSM street networks for the use as a graph, the DEM for calculating the routes under consideration of the slope, the OSM data for visualization purposes as well as the open transit data offered in the GTFS standard finally need to be adapted for the region of Carinthia. In the case of the open transit data, which is only available in Vienna and in Linz, the attributes could be changed for testing purposes.

Another step is the prototype implementation. All implemented trip planners rely on the use of an underlying graph as well as a routing engine. OSM is an open-source platform that enables the use of a street network as a graph (OSM 2013). As routing engine the OTP will be preferred, because on the one hand it enables the use of OSM data and on the other hand it is also the leading Open Source platform for multimodal trip itinerary planning and network analysis (OTP 2013). GTFS data and OSM data are basically two separate spatial layers which are only connected at the transit stops. Therefore, the implementation phase also deals with the integration of the GTFS data into the OTP engine. The core of OTP consists of three basic software components: a graph builder, a routing engine and a user interface. While the primary user interface (i.e. the map and the visual representation of the calculated route) is written in JavaScript, the other components are written in the Java programming language on the Java Enterprise Edition (Java EE) platform. The OTP routing engine is integrated into a RESTful web service API that can be used with other custom client applications (OTP Wiki 2013).
The implementation also requires the adaption of the user interface. When extending the interface with new functions like the round trip function or the consideration of the slope, the design plays an important role and therefore has to be clearly defined. An approach might be a user who enters the same start and end point for a desired route, whereas the system could immediately enable the round trip function with its options. Hochmair & Rinner (2005) also thought about a design for route planners that visualize characteristics of existing route alternatives between two selected locations before stating the route preferences. The result was that there was a high preference for various information designs like a map with bars that contain information about different routes. All these information need to be considered in the implementation phase. Finally, the results will be visualized within a web application.
An additional part of the research deals with the integration of available real-time public transit information into the trip planner, whereas two different kinds of data are available: vehicle positions and predictions about future delays. First considerations have to deal with the recording of this information to evaluate it afterwards. In the next step, in order to assess the impact of this information, the delay information at different stops has to be considered. In other words, the static timetables have to be adapted using recorded, observed timestamps from the vehicles in the network. If there is a deviation to static timetable information available, the quality of calculated routes that include real-time information is going to be evaluated. It is important to remark that the quality of these types of routes can only be determined for a period in the past, not for the future since only collected data will be considered for the route calculation.

1.4 Audience

The focus of this research is primarily on tourists, who want to plan a cycling tour, preferably in Austria as well as tourists in Netherlands that can make use of real-time public transit information. These people often do not exactly know the location of their accommodation and how to get to the next public transit station. Because of this fact, the current research could be helpful for them if they plan a trip starting directly from their housing. Another target group could also deal with residents who want to explore the country and try out new routes they didn't know before. Last but not least, a further focus could be on public transportation agencies which are not willing to publish their public transit timetables for creating innovative applications out of them. They could see the potential of this data for future developments.

1.5 Expected results

Examples like Komoot (Komoot 2013) or AnachB.at (AnachB 2013) show that there is a huge innovation potential of how to combine cycling data with additional information from different routing networks. The limitation of these platforms is the use of commercial data, which may neither be refined nor adapted by another developer. As an example, Komoot uses Google Maps (Maps 2013) data and AnachB.at uses Teleatlas (Teleatlas 2013) data for routing purposes. Using Open Data and Open Source software, this approach is very cost-efficient.

On the other hand, open real-time data in public transportation offer a great opportunity of developing new applications that inform the passenger about current delay information. This information also enables the analysis of how effective the given data to the passenger is.

All in all, the project is split into two following objectives that must be taken into account and realized:

- The creation of a criteria catalogue based on a conceptual design for evaluating the user's needs and the types of data and sources to be used.
- The conceptual modelling of multimodal data that will be used. This part also includes the data models that are going to be used for the routing graph.
- The preparation and integration of open transit data to be used in the routing engine.
- The affiliation of the gained DEM data as well as the transit data with the routing engine.
- The creation of a routable graph for Austria as well as Netherlands to be used the OTP engine.
- The implementation of a round trip function for cyclists, considering the availability of open transit data.
- The archiving and recording of real-time data for analysis purposes
- The definition of user scenarios to evaluate the differences in single route calculations, which also includes:

- The implementation of a whole analysis framework that is able to communicate with databases and the OTP routing engine
- The analysis of the impact of route planning with and without the use of real-time delay information
- The assessment of the quality and the effectiveness of real-time public transportation services

Summing up these points, the main goal is getting to know the OpenTripPlanner (OTP) and its engine, integrating public transit data for enhanced routing functionality and afterwards to expand the system by a round trip function. Finally, a further aim is also to discuss the use of real-time public transit data and how it affects the route selection. Broader details will be discussed in chapter 2 "identification of research questions" and chapter 5 "implementation".

1.6 Structure of the thesis

In the first chapter, the hypothesis for this research project is explained, which includes the scope of the project. The second chapter focuses on the research problems of both the round trip recommendation for cyclists as well as on the analysis of real-time information in public transportation. The third chapter includes a detailed state of the art as well as a literature review. Terms like public transportation timetables as well as national and international data exchange standards and models are stated. As a big part of this chapter, a detailed view about real-time standards, the routing problem and related projects is given.

In chapter four, the methodology for setting up the OpenTripPlanner (OTP), the implementation of a round trip recommendation as well as the evaluation of real-time public transit information is described.

The fifth chapter focuses on the implementation, whereas the research is split into two parts: the use of public transit data in Austria and the evaluation of the effectiveness of real-time public transit information in Netherlands.

Chapter six deals with the results of the real-time analysis and highlights the effects of the two different kinds of real-time data. In this chapter, pre-defined scenarios are also tested and evaluated. Chapter seven discusses the results and chapter eight summarizes the approach. It includes a conclusion as well as future work on this topic.

2. Identification of research questions

In this chapter, research questions will be identified and discussed. During this project, all goals will be attempted to be solved and finally discussed in the last section of this paper.
This research is separated into two independent parts: a smaller part that focuses on the combination of public transit data and bicyclers in Carinthia, as well as the main part that focuses on evaluating the effectiveness of real-time information in multimodal public transport trip planning in Netherlands.

Stroemer (2006) conducted a survey from February 2005 to February 2006, which was addressed to persons who perform bicycle tours in their leisure and found out that 60% of all people would like to have a route recommendation and more than 73% of all interviewed persons prefer a round tour instead of a simple route from A to B (67%).
These statistics show a high demand for a round trip recommendation when planning a bicycle tour. An approach for a round trip calculation for pedestrians, based on a distance input, has already been developed by Zipf & Jöst (2006) and Stroemer (2006). Only one platform, “Komoot” (Komoot 2013), that enables a round trip recommendation (without public transit information), was found during this research. Neither OTP nor AnachB.at nor Google Maps do include such a function, so there is definitely a call for further development.
In order to combine such an application with public transit data, available transit information including timetables in Austria need to be gathered. Due to the limited availability and access to this data, this task is going to be a challenge. The only providers of Open Data interfaces in Austria are “Wiener Linien” (Wiener Linien 2014) and “Linz AG Linien (Linz AG Linien 2014). The Austrian federal railways (ÖBB) already delivered their time table information as GTFS feed to Google (DerStandard 2013), but not as Open Data for developers (Kurier 2013). This agency could be a possible source for sample data.

A further research question deals with the analysis of real-time public transportation information. This kind of data is barely available to the public in Europe since it refers to a completely new technology, which is very expensive to establish in a public transit network (Roush 2012). A possible data source for this matter could be OVapi, Netherlands (OVapi 2014), which is the only provider of GTFS realtime (Google Realtime 2014) data in Europe (Transit Developers 2014a).
As this real-time standard from Google is an extension of the General Transit Feed Specification (GTFS), both data types can easily be linked. To perform the analysis, aspects like an effective concept and design of a spatial database need to be considered. The huge amount of recorded data from one day needs to be stored there to be evaluated.
A main part of the research deals with the question, if real-time information that is available during the route calculation process is an advantage for the passenger or not. To answer this question, altogether five different scenarios will be simulated using available timetables with and without the integration of real-time

information. These scenarios use two different types of passengers: one that prints the map directly after the route calculation and one that uses his or her smartphone. The main goal should show possible impacts to the route selection, if delay information is available.
A further part deals with the identification of quality issues of real-time data. Questions like the coverage and accuracy need to be answered via maps and diagrams. To evaluate the accuracy, observed arrival and departure times from recorded vehicle positions can be compared with collected estimated future delays.

For both separated parts of this research, the used multimodal routing framework will be the OpenTripPlanner (OTP). The only aspect that needs to be considered is the data source to be used as a routing graph, besides the transit network. Possible sources in Austria could be Open Governmental Data (Data KTN 2014) or Graph Integrated Platform (GIP) data (Mandl-Mair 2014). A universal data source for both parts could also be OpenStreetMap (OSM 2013).

3. State of the Art and literature review

3.1 Public transportation

In the World Wide Web, several definitions of the term “public transportation” are available. TFGM (2014) says that it includes “...travel services provided locally that allow lots of people to travel together along set routes”.
Another definition comes from UITP (2014), which explains:

> *“Public transport, public transportation, public transit or mass transit comprises all transport systems in which the passengers do not travel in their own vehicles. While it is generally taken to include rail and bus services, wider definitions include scheduled ferries, taxicab services etc. — in other words, any system that transports members of the general public.*
> *Public transport is usually regulated as a common carrier and is usually configured to provide scheduled service on fixed routes on a non-reservation basis. The majority of transit passengers are traveling within a local area or region between their homes and places of employment, shopping, or schools.”*

Most of the public transit is operated by a scheduled time table, which is adapted to departure and arrival times of different transportation modes.
There are big differences in the administration of public transit in different countries of the world, though. In Asia, most of the mass transit operations are managed by private companies, whereas in North America these operations are generally administered by municipal transit authorities. In Europe, most of the mass transit operations will be handled by outsourced private transport operators. All these services can be, depending on the different countries, either profit driven (depending on ridership numbers) or funded by government subsidies (by local or national tax revenue) (PT Wiki 2014).

3.1.1 Scheduled timetable

A scheduled timetable is a printed or electronic document that shows service periods out of public transportation modes like buses or trains. This information can help people to get knowledge about arrival and departure times from different stations to finally plan a trip. The document may consist of several movements on a particular trip, as well as latencies between trips.
Timetables are available in different representations. The most common format is the Matrix format, where you can see the particular stations in the rows, and service times in the columns. Figure 1 shows a timetable from an Austrian federal railways (ÖBB) bus route in Lower Austria, Austria.

NÖ Landesausstellung
Asparn/Zaya - Poysdorf - Mistelbach - Poysdorf - Asparn/Zaya
gültig vom 27.4. bis 3.11.2013

POSTBUS

Montag - Freitag

Asparn/Zaya Landesausstellung				9.10		10.10		11.10		12.10		13.10		14.10			15.40		16.40		17.40		18.40
Poysdorf Landesausstellung	ab				9.02		10.02		11.02		12.02		13.02		14.02	15.23		16.23		17.23		18.23	
Mistelbach Museumszentrum					9.24		10.24		11.24		12.24		13.24		14.24	15.45		16.45		17.45		18.45	
- Bahnhof	an			9.22	9.31	10.22	10.31	11.22	11.31	12.22	12.31	13.22	13.31	14.22	14.31	15.52	15.52	16.52	16.52	17.52	17.52	18.52	18.52
- Bahnhof	ab	8.33	8.33	9.33	9.33	10.33	10.33	11.33	11.33	12.33	12.33	13.33	13.33	14.33	14.33	15.55	15.55	16.55	16.55				
- Museumszentrum		8.40		9.40		10.40		11.40		12.40		13.40		14.40			16.02		17.02				
Poysdorf Landesausstellung	an	9.00		10.00		11.00		12.00		13.00		14.00		15.00			16.22		17.22				
Asparn/Zaya Landesausstellung			8.45		9.45		10.45		11.45		12.45		13.45		14.45	16.07		17.07					

Samstag, Sonn- und Feiertag

Asparn/Zaya Landesausstellung					9.10			10.10		11.10				12.10		13.10				14.10			15.23	▸▸▸
Mikulov na Morave U Parku									9.32						11.32						13.32			
Poysdorf Landesausstellung	an								9.54						11.54						13.54			
- Landesausstellung	ab								10.02		11.02				12.02		13.02				14.02	15.06		
Mistelbach Museumszentrum									10.24		11.24				12.24		13.24				14.24	15.28		
- Bahnhof	an				9.22			10.22	10.31	11.22	11.31			12.22	12.31	13.22	13.31			14.22	14.31	15.35	15.35	
- Bahnhof	ab					9.35						11.35						13.35						
Niedersulz Museumsdorf	an					9.54						11.54						13.54						
- Museumsdorf	ab						10.00						12.00						14.00					
Mistelbach Bahnhof	an						10.19						12.19						14.19					
- Bahnhof	ab	8.33	8.33	9.33	9.33			10.33	10.33	11.33	11.33			12.33	12.33	13.33	13.33			14.33	14.33	15.36	15.36	
- Museumszentrum		8.40			9.40			10.40		11.40				12.40		13.40				14.40			15.43	
Poysdorf Landesausstellung	an	9.00			10.00			11.00		12.00				13.00		14.00				15.00			16.03	
- Landesausstellung	ab	9.05						11.05						13.05						15.05				
Mikulov na Morave U Parku		9.25						11.25						13.25						15.25				
Asparn/Zaya Landesausstellung			8.45	9.45					10.45		11.45				12.45		13.45				14.45	15.48		▸▸▸

Asparn/Zaya Landesausstellung					16.23		17.23				18.23
Mikulov na Morave U Parku				15.32						17.32	
Poysdorf Landesausstellung	an			15.54						17.54	
- Landesausstellung	ab			16.06		17.06				18.06	
Mistelbach Museumszentrum				16.28		17.28				18.28	
- Bahnhof	an			16.35	16.35	17.35	17.35			18.35	18.35
- Bahnhof	ab	15.40						17.40			
Niedersulz Museumsdorf	an	15.59						17.59			
- Museumsdorf	ab		16.00						18.00		
Mistelbach Bahnhof	an		16.19						18.19		
- Bahnhof	ab			16.36	16.36						
- Museumszentrum					16.43						
Poysdorf Landesausstellung	an				17.03						
- Landesausstellung	ab				17.08						
Mikulov na Morave U Parku					17.28						
Asparn/Zaya Landesausstellung				16.48							

Fig. 1: Time schedule of a particular Austrian Postbus route

Figure 2 shows another representation of a matrix timetable which represents departure times of trains from Berlin (GER) to Munich (GER). It was obtained by Deutsche Bahn (DB).

Berlin Hbf (tief) → München Hbf

ab	Zug	Umsteigen	an	ab	Zug	an	Verkehrstage
10:40	IC 2250	Weimar Gera Hbf Regensburg Hbf	12:58 13:56 17:30	13:07 14:09 17:46	RE 3659 RE 3645 ALX 86015	19:19	So 10
10:46	EC 175	Dresden Hbf Nürnberg Hbf	12:52 18:17	13:56 19:10	RE 3788 RE 4021	20:52	täglich
10:59	ICE 1709					16:39	So 10
10:59	ICE 1509					16:39	Mo - Sa 05
11:53	ICE 1003	Nürnberg Hbf	16:20	16:27	ICE 789	17:38	täglich
11:53	ICE 1003					18:12	täglich
12:46	EC 177	Elsterwerda Chemnitz Hbf Hof Hbf	14:06 15:47 17:34	14:13 15:59 17:40	RB 17511 IRE 3090 ALX 86017	21:16	täglich
12:58	ICE 1511					18:40	täglich
13:53	ICE 1613	Nürnberg Hbf	18:20	18:27	ICE 881	19:38	täglich

Fig. 2: Time schedule of a tour from Berlin to Munich

Timetables can be printed in books[1], as folders, in posters[2], visualized on homepages, mobile apps[3] or electronic boards in central stations[4], published as Open Data (OD)[5] and even sent by Short Message Service (SMS)[6].

---

[1] http://2.bp.blogspot.com/-LcmivAO39fw/UL0kz-3p16I/AAAAAAAAflQ/7o_F-ecJ-hQ/s1600/TTcover.jpg, last access 17.01.2014
[2] http://upload.wikimedia.org/wikipedia/commons/3/31/FahrplanTiefenbrunnen.JPG, last access 17.01.2014
[3] http://techgeek.com.au/wp-content/uploads/2012/05/sydneypublictransport6.png, last access 17.01.2014
[4] http://upload.wikimedia.org/wikipedia/commons/c/c6/Portland_Mall_digital_sign.jpg, last access 17.01.2014
[5] http://data.gov.uk/dataset/traveline-national-dataset, last access 17.01.2014

3.1.2 Open Data

The term “Open Data” was previously described in the motivation section of the first chapter. In this chapter, this expression is linked to term public transportation, which is thought to be provided by the public transport agencies.
Kaufman & Wagner (2012) describe “open” as:

> *“…an organization must make available to the public internal data in a format usable by both interested individuals and application programmers. The opening of data should generate improved communications between transportation organizations and their customers, resulting in improved travel and services. The benefits of opening up data include more efficient travel (with an enhanced ability to find optimal routes while on the go), a greater understanding of finance/administration (helping to possibly promote improved funding), and crowdsourced analysis capabilities (potentially helping detect schedule improvements or errors in stop locations/names, for instance)…The data, which would typically include sets like schedules, routes, budgetary information, ridership numbers, traffic numbers and road conditions, should be released in both historic and real-time for both analysis and prediction. The data should be released closest to its most original format, barring security-sensitive inputs, and regardless of potentially negative concerns (which are typically neutralized by positives in other data). Being open involves sharing information for optimized travel, management, and future improvements.*

Releasing data has several benefits for the agencies: they guarantee a free development of applications. Because of the fact that the apps support different languages, it leads to increased ridership. More accurate applications satisfy customers and supply an improved customer service. Finally, a huge benefit for the agencies is the positive image gained by the public.

Altogether, the benefits can also be split up into four different entities (Figure 3). First, the benefit to the users, which are for instance always aware of traffic jams. Second, the benefit to the businesses, which could use waiting times for increasing their sales. Third, an improved public understanding of funding and why each passenger has to pay exactly that price. And last but not least, the benefit of improved transportation management data that can be used for adjustments in the schedule.

[6] http://www.oebb.at/de/Reiseplanung/Fahrplanauskunft/Mobile_Dienste/SCOTTY_SMS/, last access 17.01.2014

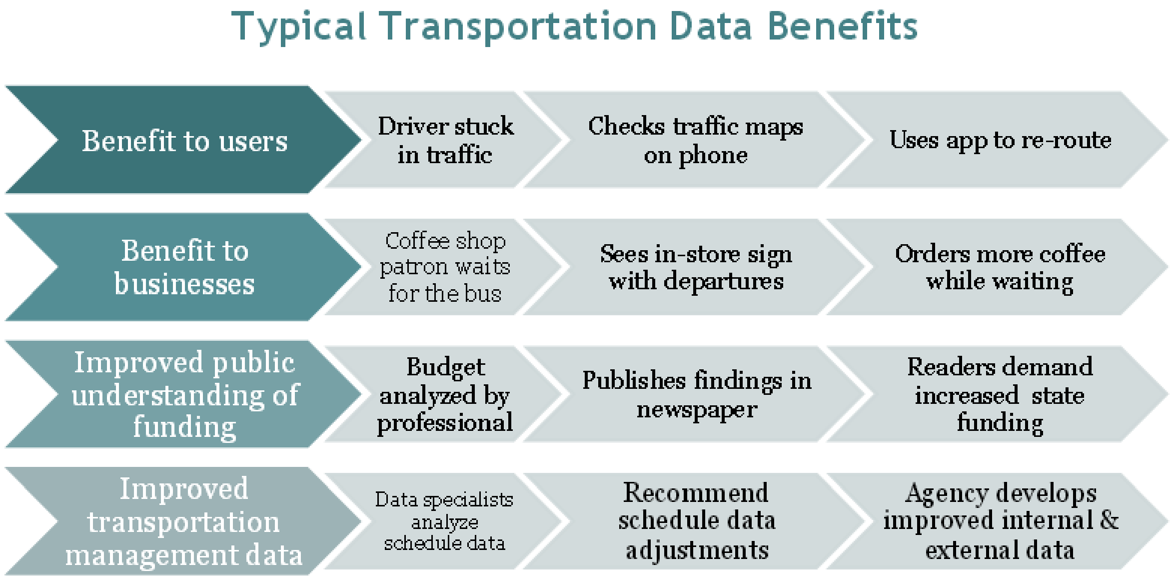

Fig. 3: Typical transportation data benefits to different participants (Kaufman & Wagner 2012)

There are two barriers of providing transportation data to the public: first of all the collection and maintenance of these data is very difficult and costly, and on the other hand it is hard to create a commercial business out of it because no one will pay for such a service (Lyoen et al 2010). Most transit information is locked into proprietary formats and systems and cannot be easily shared, viewed, updated or co-mingled without permission from the vendor and expert data analysis (Kelly et al 2011).

Nevertheless, to make aware a communication between different transport systems, standards become more and more important. Soares & Martins (2013) describe transport systems as:

> *"...distributed systems with very complex information requirements. So, the full interoperability of these systems can only be achieved through the existence and implementation of adequate standards, properly conceived by experts, tested and understood by practitioners. Strong standard for transportation data are important for the safe and efficient operation of the systems.*
> *In the last decades, several international projects and international standardization bodies have gathered efforts to develop a set of basic standards and procedures to ensure the interoperability of public transport, enabling the effective sharing of information between the different transport systems."*

3.1.3 Public transportation standards

In the following Table 1, different public transportation data standards and formats are listed. To conclude, the list is not complete, so there are even more standards in different countries all over the world. All data standards from the first half of the table can be described by one of the formats in the second half.

Table 1: Transit data standards from all over the world & their appropriate data formats. Information from Kaufman & Wagner (2012), DFT (2013), FDGC (2014), Raschke (2011)

SIRI…Service Interface for Realtime Information, NeTEx…Networks and Timetables Exchange, IFOPT…Identification of Fixed Objects in Public Transportation, DATEX… Data Exchange, NaPTAN…National Public Transport Access Node, NPTG…National Public Transport Gazetteer, VDV…Verband Deutscher Verkehrsunternehmen, CSV…Comma Separated Value, TXT…Textfile, GIS…Geographic Information System, KML…Keyhole Markup Language, XML…Extended Markup Language, FGDC…Federal Geographic Data Committee.

NAME	ORIGINATOR	WHERE IT'S USED	PURPOSE	MORE INFORMATION
		DATA STANDARDS		
GTFS	Google	World-wide	Scheduled data	https://developers.google.com/transit/gtfs
GTFS-realtime	Google	Selected US & Europe-	Real-time data	https://developers.google.com/transit/gtfs-realtime/
SIRI	European Committee for Standardization	EU	Real-time data	http://user47094.vs.easily.co.uk/siri/
TransModel	French Ministry of Transport	EU	Model of common public transport concepts	http://www.dft.gov.uk/transmodel/
NeTEx	European Committee for Standardization	EU	Exchange of public transport schedules	http://www.vdv.de/netex.aspx
IFOPT	European Committee for Standardization	EU	NaPTAN concept for European Union	http://www.dft.gov.uk/naptan/ifopt/
DATEX 2	European Commission	EU	Traffic data & management	http://www.datex2.eu/content/datex-background
TransXchange	UK department of Transport	UK	Bus schedules & data	http://www.dft.gov.uk/transxchange
CycleNet XChange	UK department of Transport	UK	Cycle path data of different communities	http://www.dft.gov.uk/cyclenetxchange/
NaPTAN	UK department of Transport	UK	Points of access to public transport	http://www.dft.gov.uk/naptan/
NPTG	UK department of Transport	UK	Topographic database of towns and settlements	http://www.dft.gov.uk/nptg/
JourneyWeb	UK department of Transport	UK	Provide multimodal journey planning	http://www.dft.gov.uk/journeyweb/
Transportation: Transit	FGDC	US	Exchange of transportation data related to transit systems	http://www.fgdc.gov/training/nsdi-training-program/materials/framwork-training-transportation-transit.pdf
NextBus	NextBus Inc.	US	Real-time data	http://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf

railML data exchange format	ERIM	EU	exchange of railway network data	http://www.railml.org/tl_files/railML.org/documents/science/270913_trafIT_FinalReportFeasibilityStudyRailTopoModel.pdf
VDV Real Time Interfaces 353/354	VDV	GER	Real-time data	http://mitglieder.vdv.de/en/wir_ueber_uns/vdv_projekte/siri.html

DATA FORMATS

CSV	Many	World-wide	data tables	http://www.ehow.com/how_5091077_use-csv-files.html
TXT	Many	World-wide	Text files	http://en.wikipedia.org/wiki/Text_file
GIS	Many	World-wide	Geographic mapping	http://en.wikipedia.org/wiki/GIS_file_formats
KML	Google	World-wide	Google Maps & Earth	https://developers.google.com/kml/documentation/
XML	Many	World-wide	Large data sets	http://www.w3schools.com/xml/xml_whatis.asp

The scope of a standard is mostly not worldwide, as in every Union and even in countries there are own departments for standardization (US vs. EU, respectively UK vs. EU although the UK is member of the EU). One of the first standards that can be globally used for the exchange of public transit data is GTFS, which is described in the next section.

The European Committee for Standardization (CEN) is an agency that is trying to achieve the needs between consumers and the industry. The established standards are concurrently national standards in each of its 33 member countries (CEN 2014). The CEN/TC 278 has more than 10 Working Groups (WG), where different standards are prepared. In the WG3 (Figure 4), also collaboration exists with International Standardization Organization (ISO) TC 204 - Intelligent Transport Systems, which is the world responsible for the development of international standards (Soares & Martins 2013).

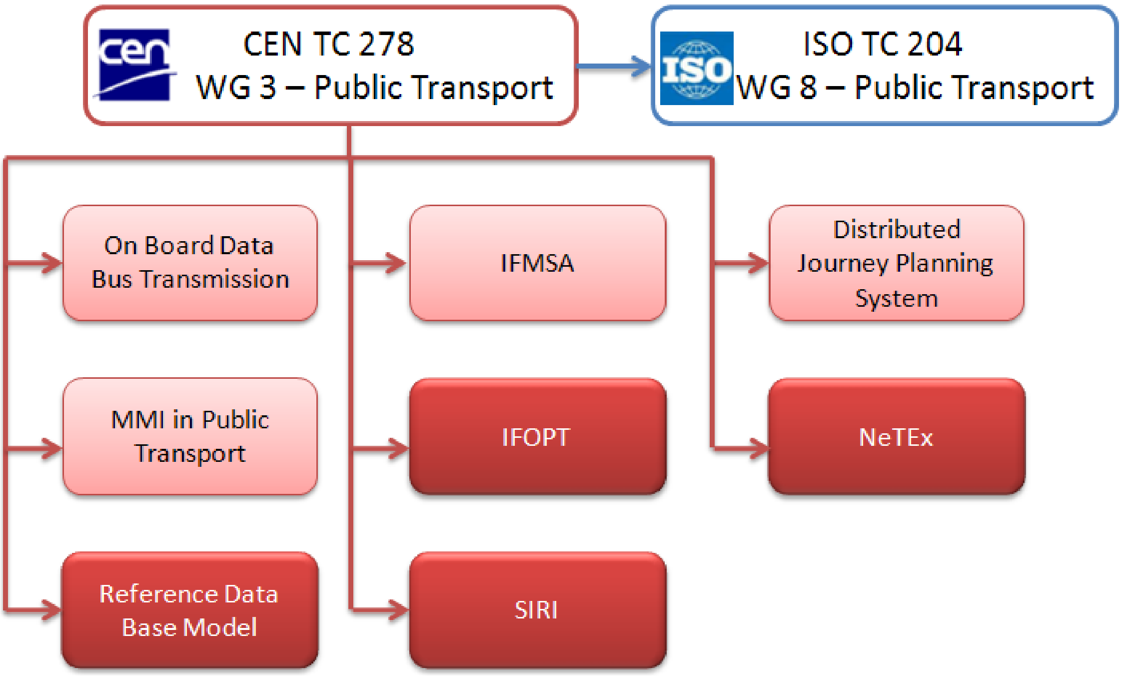

Fig. 4: Prepared standards of European Committee for Standardization (CEN) & International Standardization Organization (ISO) working groups (Soares & Martins 2013)

CEN…European Committee for Standardization, WG…Working Group, ISO…International Standardization Organization, MMI…Multimedia Interface, IFMSA… Interoperable Fare Management System Architecture, IFOPT…Identification of Fixed Objects in Public Transport, SIRI…Service Interface for Realtime Information, NeTEx…Network and Timetables Exchange

The Reference Data Base Model (Transmodel) is a data model that was developed to simplify complex information systems architecture from public transport operators (Soares & Martins 2013). It should establish a standardized communication between all entities as well as other agencies (Figure 5).

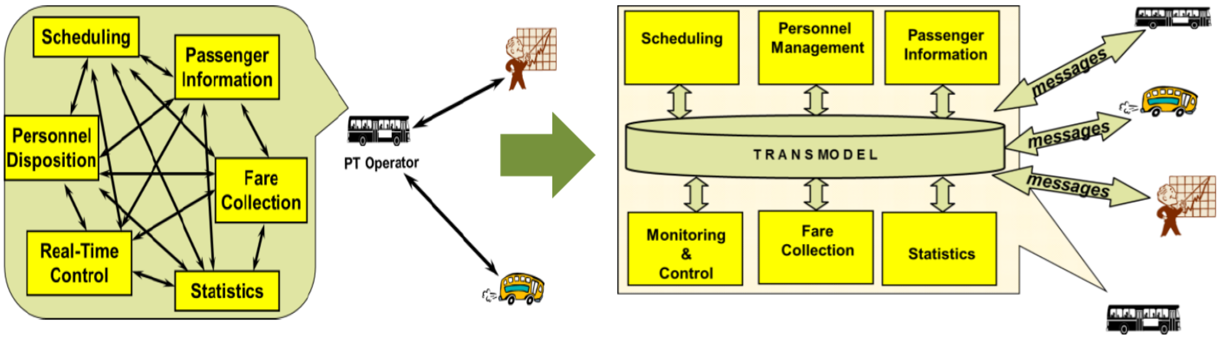

Fig. 5: Concept of Reference Data Base Model (Transmodel) data model (Soares & Martins 2013)

Service Interface for Real-time Information (SIRI) is a XML protocol that allows different public transport services and information systems to exchange real-time information. It is based on the Transmodel standard and includes timetable services as well as monitoring services. It has been established by the CEN group with a major input of the German VDV Real Time Interfaces 453 and 454 (VDV 2014a). The German standard is based on simpler structures than SIRI and due to the reduction of optional data elements it has proven to be more efficient for German purposes (VDV 2014b).
Identification of Fixed Objects in Public Transport (IFPOT) also uses the Transmodel for describing fixed objects, which are related to public transport like stop points and areas, stations, connection links, entrances, etc.
The Network and Timetables Exchange (NeTEx) is an XML schema which is based on the Transmodel, IFOPT and SIRI. It is built to relieve the communication and exchange of stops, routes and timetables between computer systems and can be seen as a complement to the SIRI standard (Soares & Martins 2013).
The idea of the railML data exchange format is to uniquely exchange railway network data from one source to another and has been established by the International Union of Railways (UIC). Due to extremely high costs in the exchange, redundancy and misinterpretation of infrastructure data, the development of such a standard could help making savings without major investments (RailML 2014).
The exchange format is described by a topological (UIC RailTopoModel) and should guarantee the scalability, sustainability of business needs as well as univocal dimensions and quality for data exchanges (railML 2013a).
In Figure 6, a possible ideal future situation is represented. It describes the data flow from different Infrastructure Manager's (IM) topological models using the railML exchange format. This allows a standardized communication between IMs among themselves.

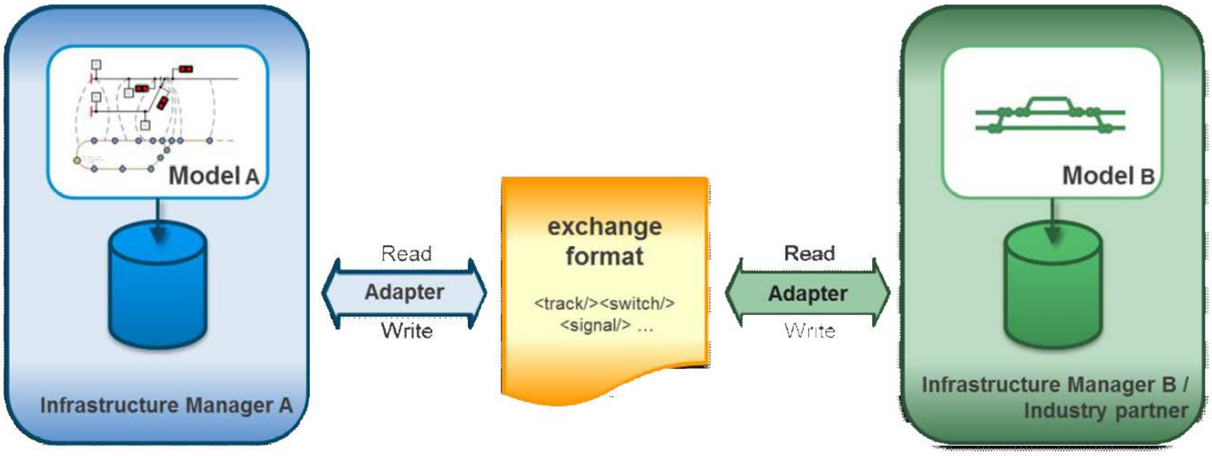

Fig. 6: Conceptual data exchange model with its requirements (railML 2013b)

3.1.4 Usage of GTFS in public transit

Jeremy Faludi (Faludi 2005) already addressed the need for the combination of interactive online maps together with online-trip planners for public transportation in 2005.

> *"The former are user-friendly and graphical, but only relevant for driving (or sometimes bicycling); the latter let you plan transit, but are user-unfriendly, because the user cannot see where any of the pick-up and drop-off points are."*

In an online blog, he pinpointed that there is a general demand on an open standard XML for deriving directions in and out of transit planning systems. His first concept was very simple:

> *"...just something to read addresses and times and tag them as get on / get off, tags to color-code segments of your trip as being bus / walking / subway / etc, and perhaps tags to include multiple alternative routes for the same trip, or tags to pass messages back and forth when the transit-planner asks you to clarify an address. This way you type your start and end addresses into Google Maps, it throws the data to the appropriate region's transit system, the transit system figures the route(s) and throws some XML back to Google, which then draws you a map with directions (and maybe includes links for alternate routes, which are served up from the same XML file). Then after an initial slightly-painful period of rewriting the trip planners to read/write this XML, and a painless little mapping-software extension like the ones listed above, all transit agencies can have fantastic user-interfaces with no effort (or future maintenance costs associated with interface). And, as a helpful consequence of universal open standards, different transit systems (say, local bus systems and city-to-city trains) could intercommunicate as well. This would solve another large user-interface hassle with transit."*

It was mid-2005, when Bibiana McHugh (McHugh 2013), an IT manager of Geographic Information Systems (GIS) for Portland's Tri-County Metropolitan Transportation District of Oregon (TriMet) agency, was wondering that even the big players of online routing platforms like Google Maps, MapQuest and Yahoo were not using public transit data for online trip planning. The kick-start was triggered by an online article of Faludi (Faludi 2005), who called the need for an open standard in the public transit field. In collaboration with Google and TriMet transit data, a prototype of Google Transit (Google Transit 2014) has been fabricated. This prototype worked with exported schedule data as CSV files. This "first widely used transit data standard", as McHugh says, was shared with Google and also to the public, so that any third-party developer could access and work with it.
Finally, on December 7^{th}, 2005, Google Transit was launched with its first release using the transit data of Portland metro area (Garg 2005). After that, Google was interested to cooperate with other transit agencies to develop this new standard format, called General Transit Feed Specification (GTFS) (McHugh 2013). The advantage of bringing the specification into CSV files is because of its simplicity and the low barrier for agencies to work with. Due to Hughes (2007), this format is:

> *"...easy to view and edit using spreadsheet programs and text editors, which is helpful for smaller agencies. It's also straightforward to generate from most programming languages and databases, which is good for publishers of larger feeds."*

In the following years, Google launched even more and more cities. Until now, more than 100 cities of all continents all over the world have already been covered by Google Transit (Transit Cities 2014).

Due to the open communication in the World Wide Web, like forums, groups as well as data exchange sites like the GTFS Data Exchange (2014), the GTFS format increased in its popularity. Agencies that supported the industry with the creation and maintenance of GTFS data began to fill an important void, so this was the time where business was initially created (McHugh 2013). McHugh also states that:

> *"The biggest advantage of being part of the GTFS standard for agencies is that their information appears in a global set of search products that are easy to use and visited by millions and millions of people every day. People who do not know a city well, are visiting, or are simply unaware of the agency's services, can benefit and find alternatives to driving. Regular public transit riders benefit from being able to find transit information in a familiar user interface and in the context of other useful information. It's about providing better information and service delivery for citizens, which is ultimately aligned with any agency's mission."*

The GTFS data can be updated frequently and published for developers to build and adapt apps. In the first time there was a general skepticism from agencies towards the name “Google” in the specification, because they didn’t want to be perceived as giving their transit information exclusively to Google. Because of this, Google changed the name of the specification to “General Transit Feed Specification”, which was recognized very positively by the transit agencies.
Nowadays, GTFS is so widely used all over the world and the default format in most communities, that it has become the “de facto” data standard in public transit data all over the world (Kaufman & Wagner 2012).
Each GTFS data feed consists out of CSV files, which are stored in a single ZIP-file. The files are listed in the following Table 2:

Table 2: Elements of a GTFS feed compared with CEN Transmodel Concept (Kizoom & Miller 2008)

GOOGLE FILE	USAGE	CONTENTS	CEN TRANSMODEL CONCEPT
agency.txt	required	Information about the transit agency.	OPERATOR AUTHORITY
stops.txt	required	Information about individual locations where vehicles pick up or drop off passengers.	SCHEDULED STOP POINT, STOP PLACE (IFOPT), TARIFF ZONE
routes.txt	required	Information about a transit organization's routes. A route is a group of trips that are displayed to the rider as a single service.	LINE
trips.txt	required	Information about scheduled service along a particular route. Trips consist of two or more stops that are made at regularly scheduled intervals.	VEHICLE JOURNEY
stop_times.txt	required	Lists the times that a vehicle arrives at and departs from individual stops for each trip along a route.	Call PASSINGTIMES STOP POINT IN JOURNEY PATTERN (SERVICE PATTERN) (ROUTE LINK - distance)
calendar.txt	required	Defines service categories. Each category indicates the days that service starts and ends as well as the days that service is available.	DAY TYPE PERIOD, DAY OF WEEK
calendar_dates.txt	optional	Lists exceptions for the service categories defined in the calendar.txt file.	OPERATING DAY
fare_attributes.txt	optional	Defines fare information for a transit organization's routes.	FARE ELEMENT PRICE
fare_rules.txt	optional	Defines the rules for applying fare information for a transit organization’s routes.	FARE ELEMENT, DISTANCE MATRIX
shapes.txt	optional	Provides rules for drawing lines on a map to represent a transit organization's routes.	ROUTE ROUTE LINK / PROJECTION
frequencies.txt	optional	Provides the headway (time between trips) for routes with variable frequency of service.	(Frequency)
transfers.txt	optional	Provides additional rules for making connections between routes.	CONNECTION LINK, SERVICE JOURNEY INTERCHANGE, SERVICE JOURNEY PATTERN INTERCHANGE, DEFAULT INTERCHANGE

3.1.5 GTFS-realtime

GTFS-realtime is an extension of the original feed and enables the update of public transportation data and schedules in an immediate way. It is an open data format for public transportation and was designed for good GTFS interoperability with a focus on quality information to passengers. The updates include live departure and arrival times of different transportation modes as well as service alerts in case of unforeseen events. The specification was introduced in 2011 by partner agencies of Google and transit developers (Google Realtime 2014).

A big difference between GTFS-realtime and the European SIRI standard is the communication model. While SIRI offers further capabilities for the integration of real-time information with operations, GTFS-realtime only works on a strictly one-way communication model. SIRI allows the communication of buses among each other for example to ensure that the passengers reach the linked bus in case if the other one is running late. It also allows the utilization of performance information to operational history and other management systems. Because of this, SIRI is primarily developed for intra-agency interoperability. The Google standard, which is used for worldwide and open communication, focuses its purpose on a simple publication of transit information from the agency itself, to be used for external bulk consumption. This open data model has activated several conservative agencies from keeping data within intra-agency networks to sharing the information outside agency walls (Reed 2013).

The specification provides three categories of real-time information (Google Realtime 2014):

- Trip updates: delays, cancellations and changed routes
- Service alerts: temporarily moved stops, unforeseen events affecting a station, route or the entire network
- Vehicle positions: information about the vehicles including location and congestion level

These categories are tended to cover most of the transit information that a passenger is interested in when using a service. The three types are stored in so-called Protocol Buffers (PB), which is Google's language-neutral, platform-neutral and extensible mechanism for serializing structured data (Google Developers 2014a). This type of streaming guarantees continuous up-to-date information. To be able to compile these files for reading and writing purposes, a "gtfs-realtime.proto" file is used (Google Developers 2014b). This "proto"-file defines the structure of realtime feed entities as well as the buildup of the three different categories trip updates, service alerts and vehicle positions. The advantage of the GTFS-realtime specification and the protocol buffer format is the expandability to include new capabilities, which enables the improvement and also the adaption of agency-specific features (Google Developers 2014c).

The integration of this information allows the development of many new applications and additionally increased the reputation of the GTFS standard all over the world.

Reed (2013) states that another factor in the growth of GTFS-realtime deals with the availability of so-called “repeaters”, which are able to convert data from different specifications to GTFS-realtime. This is very helpful to agencies that operate with real-time passenger or vehicle information in a certain format of another standard and want to benefit from an open standard like GTFS-realtime.
The static GTFS specification is already used by hundreds of agencies all over the world and can easily be enhanced in the future. An argument of Reed (2013) puts the advantage in a nutshell:

> *“Because the GTFS-realtime feed works in conjunction with GTFS, it stands to reason that many agencies will invest in making their schedule information work seamlessly with their real-time information. Applications that deliver real-time information along with scheduled information (e.g., to provide information on route geometries and stop locations along with real-time arrival times) require the reconciliation of object identifiers in schedule and real-time systems. In other words, trip identifiers or route identifiers in the schedule must match (or be translated to match) those identifiers in AVL systems. Nevertheless, GTFS and GTFS-realtime appear to be in a strong position to serve that role…”*

In 2011, when this specification was launched, six cities (Boston, Portland, San Diego, San Francisco, Madrid and Turin) were chosen to deliver their data to Google Maps. So, if an icon to select a public transit stop is clicked, live information about the departure and arrival time is displayed to the user (Roush 2012). Regarding the access to this data, it is to mention that in Europe the real-time feeds from Madrid and Turin are not accessible by the public. Over the past years, a Dutch public transit agency called OVapi introduced GTFS-realtime in its organization, which is the only company in Europe that provides Open Data (OD) access to their live transit information. They introduced the access to the realtime feeds at the end of January, 2014 (Transitfeeds 2014, Transit Developers 2014a). A possible reason why other transit agencies are not willing to incorporate is because of the enormous costs of the system and managing the whole equipment (Roush 2012). Although there are other companies all over the world that introduced the specification, the focus of this research is in Europe. A short overview of some particular agencies is given in the following Table 3:

Table 3: Overview about transit agencies that offer GTFS/GTFS-realtime data. VBB…Verkehrsverbund Berlin-Brandenburg, BART…Bay Area Rapid Transit, TriMet…Tri-Country Metropolitan Transportation of Oregon, MARTA…Metropolitan Atlanta Rapid Transit Authority, EMT…Empresa Municipal de Transportes, GTT…Gruppo Torinese Trasporti

NAME	GTFS / GTFS-REALTIME	OPEN DATA?	HOMEPAGE
Mecatran (Montpellier, FRA)	GTFS-realtime	only Google Maps	http://www.mecatran.com/en/content/mecatran-launch-first-gtfs-real-time-feed-france
VBB (Berlin, GER)	GTFS	Open Data	http://daten.berlin.de/kategorie/verkehr
InterConnex (Berlin, Leipzig, Warnemuende GER)	GTFS	Open Data	http://www.gtfs-data-exchange.com/agency/interconnex/
BART (Oakland, USA)	GTFS-realtime	Open Data	http://www.bart.gov/schedules/developers/gtfs-realtime
TriMet (Oregon, USA)	GTFS-realtime	Open Data	http://developer.trimet.org/GTFS.shtml
MARTA BUS (Atlanta, USA)	GTFS-realtime	Open Data	http://www.itsmarta.com/developers/data-sources/marta-bus-gtfs-realtime.aspx
EMT (Madrid, SPA)	GTFS-realtime	only Google Maps	http://lists.kde.org/?l=kde-commits&m=135430787715412&w=1
GTT (Turin, ITA)	GTFS-realtime	only Google Maps	http://opendesktop.org/content/show.php?content=155821
OVapi (Rotterdam, Amsterdam, NED)	GTFS-realtime	Open Data	http://gtfs.ovapi.nl/nl/

3.1.6 GTFS/GTFS-RT vs. VDV452/453/454

GTFS is a worldwide data standard that allows the exchange of transit information in a simple manner. Although it was invented by Google under consideration of integrating transit data into their own application “Google Maps”, it is also a common format for general exchange purposes and tends to be used outside the organization. GTFS is designed just for customer information and doesn’t include additional agency information. Due to this fact, it is a very slim and strict format that only contains essential information. In contrast to GTFS, VDV interfaces consist of a major variety of agency information as well as enhanced details for customers (Bleyl & Frommenwiler 2012). It was established by the “Verband Deutscher Verkehrsunternehmen” (Association of German Transport Operators – VDV) for the creation of standardized data interfaces for the ÖPNV (Öffentlicher Personennahverkehr) data model (VDV Schriften 2013). The purpose of VDV interfaces is the efficient and standardized use of data inside an agency and the exchange of data between agencies, but not considered for the public scope. Due to the increasing demand of open public transport data and also the very high market share of Google in this field, VDV interfaces 452/453/454 are considered not to play any role in the future (Zukunft Mobilitaet 2014). In Table 4, the difference between VDV and GTFS information is described:

Table 4: Difference between the VDV interfaces and the GTFS feeds. Information from VDV (2014b), VDV (2014c) and VDV (2014d) AVMS...Automatic Vehicle Management System

	VDV TYPE	TYPE OF INFORMATION	PURPOSE	GTFS TYPE	TYPE OF INFORMATION	PURPOSE
timetable – information	VDV 452 "Network / Timetable"	calendar data	data types and their validity in the company calendar	GTFS	calendar.txt	service categories that indicate the days of service
		operational data	vehicle stock, vehicle types, announcement texts and destination texts		calendar_dates.txt	exceptions for service categories
		location data	bus stops, stopping points, beacons, depots		stops.txt	locations stopping points
		network data	route sections, distances, running time groups, running times, stopping times		routes.txt	transit organization's routes; group of trips
		line data	lines and courses for different routes		trips.txt	scheduled service along a route; consists of two or more stops
		timetable data	runs and run-dependent stopping times, blocks		stop_times.txt	departure and arrival times of individual stops
		connection information	validity information about connections e.g. between journey planning system to an AVMS		transfers.txt	rules for making connections between routes
		spatial partitioning	administrative units, tariff areas		agency.txt	information about transit agency
					shapes.txt	rules for visualizing routes
					frequencies.txt	time between trips
					fare_rules.txt	rules for applying fare on routes
					fare_attributes.txt	fare information for routes
realtime – information	VDV 453 "connection reliability"	Reference Data Service for Connection Protection (REF-CP)	exchange of planned schedules for connection protection	GTFS-RT	alerts.pb	stop moved, unforeseen events affecting a station, route or the entire network
		Process data service for connection protection (CP)	exchange of real-time data for connection protection		tripUpdates.pb	delays, cancellations, changed routes
		Reference Data Service for Passenger Pnformation (REF-DPI)	exchange of location related planned schedules for passenger information		vehiclePositions.pb	information about the vehicles including location and congestion level
		Process Data Service for Passenger Information (DPI)	exchange of real-time data for passenger information		gtfs-realtime.proto	hierarchy of elements and their type definitions, used for compilation of RT feed
		Process Data Service for Visualisation (VIS)	exchange of real-time data for the visualisation of vehicles in foreign control centres			
		General Message Service (GMS)	exchange of textual information between the control centres			
	VDV 454 "dynamic passenger information"	Reference Data Service for Schedule Information (REF-SIS)	exchange of planned schedules for schedule information			
		Process Data Service for Schedule Information (SIS)	exchange of real-time data for updating the schedule information with current data			

3.2 Routing problem

Routing is a method of determining the optimal path in a network. While the term “routing” can be used for different kinds of networks, e.g. when transferring data packets through the internet or sounds through the telephone network, it is linked to transportation networks in this research. For this purpose, this process is often calculated with a shortest-path algorithm to find the fastest route (Elias & Hampe 2003), but can also deal for instance with the computation of the most attractive route, especially for cycling purposes (Hochmair & Navratil 2008) or even for round-trips (Stroemer 2006).

A trip planner requires a set of features to calculate a route from an origin to a destination. First of all, a policy to enter an origin and a destination, which is often done by either selecting the point on a map or typing the name of the desired place, is required. Secondly, a routing graph is needed which consists of a street network and further information about restrictions and directions. Furthermore, a routing algorithm is required, which should be able to find a path from the starting point to the end point. The calculation may prioritize several vertices and edges of the street network in order to fulfil the user’s preferences. Another requirement is a procedure to set the user’s preferences, depending on the mode of transportation he wants to use. Finally, all the results need to be visualized, so there needs to be a method to show all necessary information to the user (Kelly et al 2011).

In the following model (Figure 7) you can see a graphical overview of a routing system with all its different parts (Bronsveld 2012).

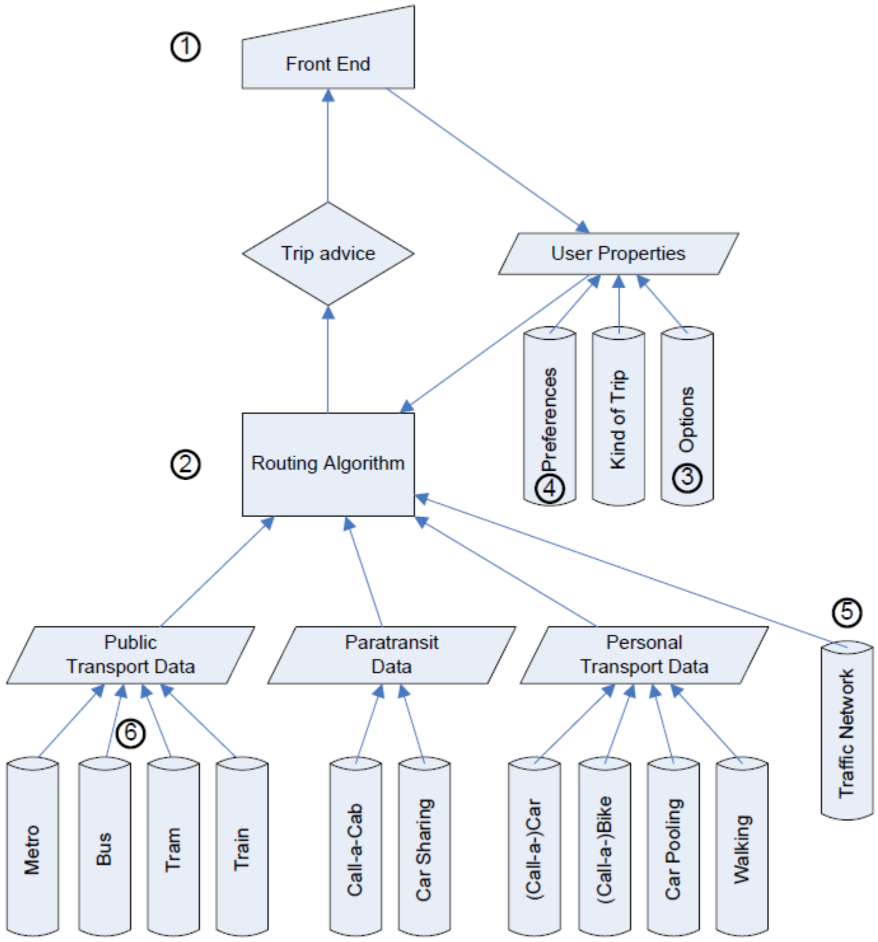

Fig. 7: Overview about major components of a routing system (Bronsveld 2012)

The Front End (1) is an interface that allows the user to easily interact with the product by defining or entering search parameters, for instance.
When typing a desired origin and destination into the search field of a trip planner, the search is often linked to a database that includes the geocoded names with their location in the real world. When a desired place or address is found, the system informs the user by setting a pin point on the Place of Interest (POI), which can also be adjusted by drag-and-drop. Another possibility includes the use of a Global Positioning Sensor (GPS) directly in the hardware of the used device. It allows an automated positioning of the user's location, which can be used either as start or as end point of the desired route.
The routing algorithm (2) is the core of the trip planner and calculates the route by the use of vertices, edges and their weights. Weights will be changed by using different modes of transportation, where the shortest path is defined by the path with the lowest cumulative weight (Kelly et al 2011). A classical implementation of a shortest-path algorithm is called "Dijkstra". It is a static routing algorithm, where the weights of the edges do not change during the computation. The "Dijkstra" is looking at the nodes of the road network in order of their distance to the source node, and then calculates the route between origin and destination (Delling et al 2009).

Another algorithm is called "Greedy Best-First-Search", which works in a similar way like the "Dijkstra", except that it has some heuristic approach. Instead of selecting the vertex closest to the starting point, it finds the vertex closest to the goal. A further algorithm, "A*", is the most popular choice for path finding because it combines the information of "Dijkstra" together with "Greedy Best-First-Search" (Stanford 2014).

In its recent version, the OTP uses the "A*" algorithm and "Contraction Hierarchies". The idea behind Contraction Hierarchies is the better performance in large graphs with non-transit segments. For additional routing performance in transit routing, an improved "A*" heuristics will be used (Kelly et al 2011).

The user properties in the Figure 6 include Options (3) and Preferences (4), whereas Bronsveld (2012) differentiates between general questions like "Is the user in possession of a frequently user pass for the Public Transport?" (3) and personal questions like "Is the user claustrophobic and is he not willing to use the metro?" (4). The traffic network (5) includes the routing graph, which consists of nodes and edges that will be connected to these nodes. Each edge has weights (e.g. speed limits, turn restrictions, traversal permissions, transition points, etc.), which are able to influence the routing engine (Kelly et al 2011).

Public transportation data (6) include all information about the different transportation modes and can be derived by standardized files, like GTFS data, for instance.

3.3 Existing data sources for routing

In order to build a trip planner, an appropriate network (street, train, metro, bike, etc.) needs to be used as a routing graph. Existing free available and commercial data sources include OSM (OSM 2013), GIP (Mandl-Mair 2014), Teleatlas[7] or Navteq[8], whereas OSM doesn't provide a routing graph itself, but only the access to the data for creating a graph for his own (Ramthun 2012).

OpenStreetMap (OSM) has been established in 2004 after the request of a free and comprehensive digital map instead of relying on expensive licenses for the use of commercial maps. Before, the creation of maps was only performed by cartographers, geographers and scientists who collected geodata during expeditions and other scientific trips. Nearly ten to fifteen years ago, the equipment to do this was so expensive that only single companies were able to create maps for their own purposes (Dirksen 2011).

The aim of OpenStreetMap deals with the creation of a global digital map, supported by the general public. Each citizen may generate content and offer it to the whole community. Goodchild (2007) called this phenomenon Volunteered Geographic Information (VGI), which enables public participation in the creation of digital geographic data.

By 20 January 2014, the number of OSM users is more than 1.5 million worldwide. Up to this date, more than 3.78 billion GPS points were uploaded and more

[7] http://www.teleatlas.at/, last access 20.01.2014
[8] http://here.com/navteq-redirect/, last access 20.01.2014

than 2.16 billion nodes, 214 million ways and 2.33 million relations were created (OSM Stats 2014).
The following model (Figure 8) shows the architecture of OSM in a simple representation. Different components are visible and how all of these are working together.

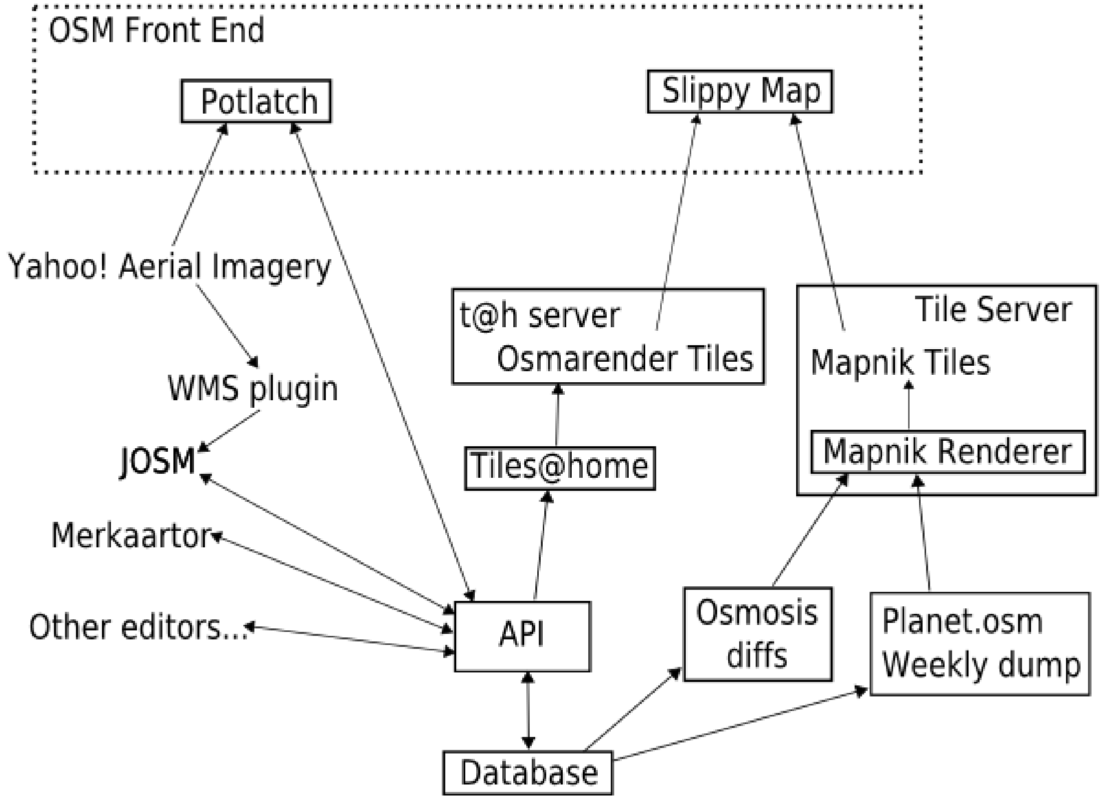

Fig. 8: Architectural design of OpenStreetMap (OSM)

All the geospatial data are stored in a PostgreSQL[9] database with a PostGIS[10] extension. This geographical information consists of nodes, ways and relations. Ways are a combination out of several nodes, whereas the relations depict complex correlations between points and/or ways and group simple objects into more complex features. Using nodes and ways, it is possible to capture both the location and the shape of an object. Examples could be a mailbox that consists out of a single node at location x and y, or a street as a way out of nodes at several locations x and y. Also, buildings or country borders can be shaped using a closed way whose start and end node is equal. To identify these geometries as real objects, so-called tags need to be captured. These tags or key-value pairs are directly attached to an element to describe it. It is possible to assign one or more tags to one element (Ramthun 2012).

[9] http://www.postgresql.org/, last access 18.01.2014
[10] http://postgis.net/, last access 18.01.2014

Each tag consists out of a key and a value. An example could be:

```
<node id="325252" lat="54.56429" lon="-2.4564" version="2" chang-
eset="3224343" user="steinerd" uid="3455" visible="true" timestamp="2014-
01-20T11:53:32Z">
    <tag k="name" v="Postbox Steiner"/>
    <tag k="operator" v="Post AG"/>
    <tag k="amenity" v="post_box"/>
    <tag k="collection_times" v="Mo-Fr 9:00,14:00; Sa-So 8:00"/>
</node>
```

As it is visible in the example before, all data are stored in an “OSM-XML” data format, which includes all ways, nodes and relations in a single osm-file.
This so-called “planet file” contains all collected geodata from all over the world and is published as a free download[11] once a week. Besides, it is also possible to download osm files for single continents or countries from the Geofabrik[12] homepage.
Osmosis is a replication that is responsible for the handling of change-sets, or in other words, to efficiently keep the copy of the planet file up-to-date (Osmosis 2014).
The Application Programming Interface (API) is used by different editing software like Java OpenStreetMap Editor[13] (JOSM) or Merkaartor[14] to edit and publish geodata. It can also be used for the implementation of own applications that use OSM data.
The tile server uses the open-source software “Mapnik” to create extracts from the map (“tiles”). The rendering process generates the “standard” OSM style and enables an export of geospatial data at the OSM[15] website (OSM Mapnik 2014).
“Tiles@home” was another rendering set-up that used an “Osmarender”[16] system, but is no longer available (OSM Slippy 2014).
In the OSM Front End, two different visualization alternatives will be offered to the user; dependent on whether he is logged in or not. If the user is logged into the system, it is possible to change and update the geospatial data directly on the website, using the Potlatch OSM editor. In contrast to JOSM which is implemented mainly for experienced users, Potlatch is also an opportunity for beginners. If the user is not logged into the system, he can only see the slippy map.
“Slippy Map” is in general a term for a modern web map where someone can zoom and pan around. This word is also used for the map that is displayed at the OSM website. The map image is built up of many tiles which were rendered using the Mapnik renderer on the tile server (OSM Slippy 2014).

Another intermodal routing graph for Austria is called “Graphenintegrationsplattform” (GIP) and has been developed by ITS Vienna Region, Vienna, Lower Austria, Burgenland and the company Prisma Solutions as an initial project “VIP Vienna Region” (from 2006-2008). This project was initialized to combine data-

[11] http://planet.openstreetmap.org/, last access 20.01.2014
[12] http://download.geofabrik.de/, last access 20.01.2014
[13] http://josm.openstreetmap.de/, last access 20.01.2014
[14] http://merkaartor.be/, last access 20.01.2014
[15] http://www.openstreetmap.org/, last access 20.01.2014
[16] http://wiki.openstreetmap.org/wiki/Osmarender, last access 20.01.2014

bases and geoinformation systems from different public agencies that deal with traffic infrastructure (local, regional, provincial and national authorities). Since 2009 up to 2015, the initial project has been expanded to several projects (gip.at, gip.at-Erweiterungen, gip.at-Erweiterungen2 and gip.gv.at), which will be funded by the government and the climate and energy fund. The purpose of this project includes the administration of streets, as base information for model calculation, routing, cartography, etc. GIP can be used for all types of transportation: motorized traffic, public traffic and non-motorized traffic (pedestrians, cyclists). Because of the fact that this graph is intermodal, also a link between these types of transportation is included (GIP 2010). GIP will be made available free of charge to all municipalities, cities and other local and regional authorities (GIP online 2014). An example of a graph representation is shown in Figure 9:

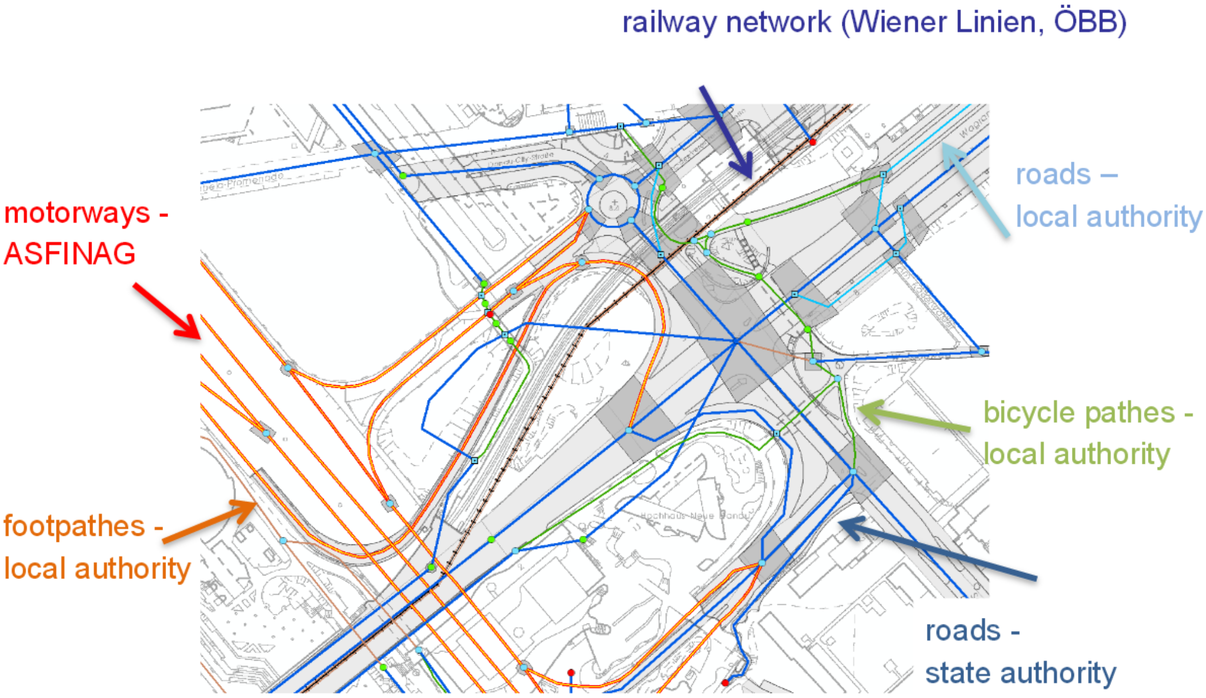

Fig. 9: Visualization of an extract from the Graphenintegrationsplattform (GIP) network (Molin 2011)

Navteq and Teleatlas are commercial data providers and global competitors in the field of digital road, street and transportation network with land cover information. As both providers are mainly providing digital map information for automotive navigation systems and are also very expensive, they are not in the scope of this project and will only briefly described.

The database of Teleatlas is called TomTom® MultiNet® and will be updated four times a year. It includes a high percentage of the landmass of Europe, the United States and Canada and also a rapidly growing portion of Africa, Asia, Australia, South America and Central America. The data are available in different data formats, incl. the ESRI® shapefile format (Teleatlas 2014).

Navteq's street data is called Navstreets® and mainly comprises the regions of the United States, Canada, Mexico and Europe (Navteq 2014).

3.4 Related projects

As this research is split into two main parts - the combination of public transit and bicycle data into a round trip recommendation, as well as the evaluation of real-time information in multimodal public transport trip planning - this chapter reflects on related projects of both of them.

3.4.1 Combination of public transit and bicycle

In existing online platforms, e.g. the interactive map of “Kaernten Werbung” (Kaernten 2013), “Radtouren.at” (Radtouren 2014) or the online cycling routing portal of “Bergfex” (Bergfex 2013), the user has to select from pre-defined routes and stored in a database, thus cannot define trip origin and destination. These websites provide filter functionalities, which allow to screen the pool of routes with regards to route characteristics by setting ranges of threshold values, e.g. for trip distance, height difference or region. It has to be mentioned that both websites “Kaernten Werbung” and “Radtouren.at” use the “outdooractive” (Outdooractive 2014) outdoor- and tours portal, which is one of the leading information source in that area. “Bergfex” primarily features tourist routes, since they are hand-picked and already include all the highlights of the certain trip. Besides, there also exist a lot of online trip planners that enable a routing option for cyclists. As an example, Google Maps (Maps 2013) is one of the largest platforms and provides a feature called “bicycling directions”, which is currently a beta version. But as Google cycling data are incomplete nowadays and Google even doesn’t combine bike with transit data, the need for another bicycle trip planner is given. Another recent system that offers a possibility to create a bike route with the intermodal GIP graph is “AnachB.at” (AnachB, 2013). It provides a real-time overview of all trains, buses or trams as well as information to get to a destination without delays. Participating entities are e.g. Wiener Linien, ÖBB, ASFINAG, the police and Ö3 radio traffic information (WIEN 2013). Another innovative platform called Komoot (Komoot 2013) allows the user to define a starting point and the amount of time that is at one’s disposal. Besides, he or she can choose between three different types of cycling routes (normal, mountain or race). The result is displayed on a map where sights located nearby are preferred. The user can either choose between an OSM or Google Maps basemap in the background.

In a further project, a team of the “Location-Aware Information Systems Laboratory” in Florida used biking and walking data from OSM together with open transit data from local agencies to set up an instance of OTP for multimodal routing (OTP Tampa 2013). The transit data are based on the General Transit Feed Specification (GTFS) standard, which has been developed by Google and is a standard for describing transit systems nowadays. They depicted the feasibility to implement a multimodal trip planner using open-source software and open data sources. The only barrier of developing such a system was the GTFS data availability and the quality of OSM data in the United States (Hilsman et al, 2011). “Fahrradies” is a routing system that allows the complete design and individual planning of a bicycle route through the tourist region “Oldenburger Münsterland” in the Northwest of Germany. Using this route planner, the user has the possibility to configure his own route, including thematic routes and selectable stops (Ehlers et al 2002).

3.4.2 Evaluation of real-time information in multimodal public transit trip planning

The analysis of real-time information has been done in several projects before. Jariyasunant et al. (2011) already evaluated the performance of a transit trip planner called “Transitr” for mobile devices. One part of this trip planner used a static database of bus schedules to calculate a route, whereas another one used estimated waiting times from a real-time bus prediction engine. For real-time planned trips, a snapshot of the actual state of the network as well as historical data were used to predict the user’s trip. When comparing the output of random routes from both calculated trip planners, differences in the accuracy of estimated travel times could be determined. They concluded that the inclusion of real-time information over schedule based transit trip planners provides marginal accurate predictions for total trip times as well as a larger number of optimal suggested routes.
Szabo et al. (2013) discuss the realization of real-time refinements to static GTFS data based on mobile participatory sensing. They make use of crowd-sensed positioning data of clients in buses to measure and improve the quality of realtime community services. As an alternative to the GTFS realtime service, this approach is very cost-effective and powerful as the number of users increases.
In Figure 10, the architecture of the Live Transit Feed Service visualizes the participating entities. The Extensible Messaging and Presence Protocol (XMPP)-Server collects raw position information from participatory users with their mobile client. The GTFS Emulator & Analytics synchronizes the collected information with data from the GTFS database and offers either guiding contents to different routes that will be used by different buses (trips), or synchronizes live events directly with single trips that will be considered during route calculation.

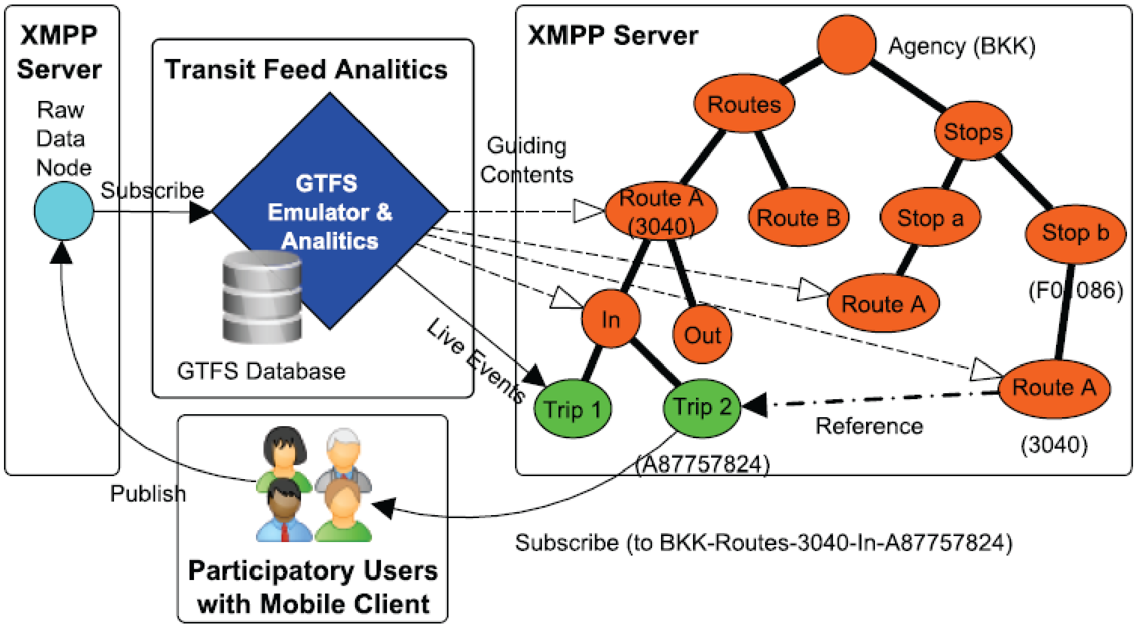

Fig. 10: Live Transit Feed Service Architecture (Szabo et al (2013))

3.4.3 Comparison of different routing platforms

A variety of online multimodal trip planners have been developed over the past years, each with different trip choice options. Portland's public transit agency, TriMet, implemented together with OpenPlans an OTP trip planning system for Portland and opposed the results from fifteen single trips. The tests identified visible improvements of the new software, compared to TriMet's older, proprietary solution, the Proprietary Transit Trip Planner (PTTP). In Figure 11, the results of a bike-to-transit trip analysis are visualized. When OTP bike-to-transit trips were compared against single-mode transit trips in Google Transit and PPTP, an enormous capacity in travel time savings could be achieved (McHugh 2011). This output shows that there is a huge optimization potential when using a multi-mode trip planner instead of a single-mode trip planner. This statistic also shows that using the OTP routing engine for this research is the best choice.

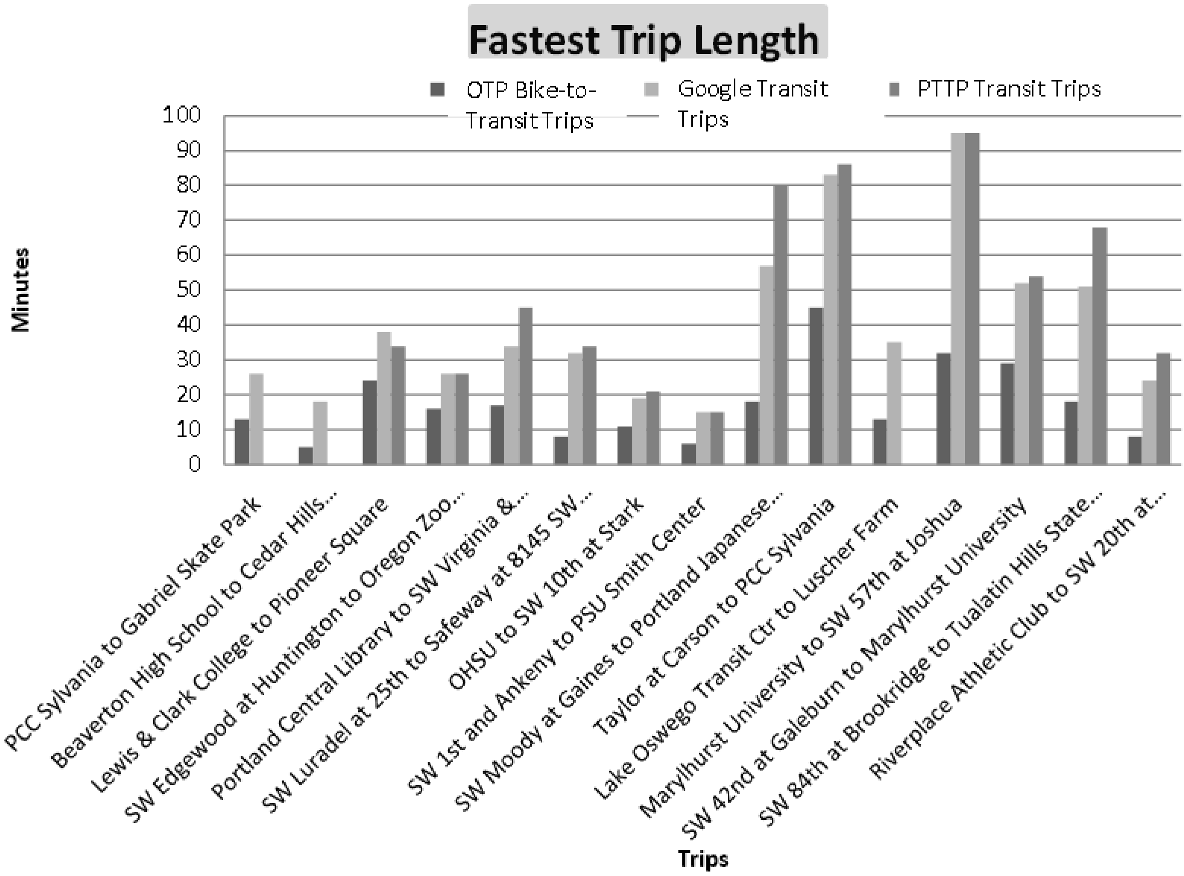

Fig. 11: Comparison of three different route planners: OpenTripPlanner (OTP), Google Transit and TriMet's Proprietary Transit Trip Planner (PTTP) (McHugh 2011)

4. Method of solution

In this chapter, the methodology for setting up the OpenTripPlanner (OTP) as well as the transformation of the GTFS feed is described. A general overview of use cases for the trip planner is given in Figure 12. A more detailed view is represented in the data flow model to see the dependencies between different actors and the transfer of data (Figure 13). In the transit data conversion section, the activity of transforming scheduled time table into the GTFS format will be discussed. Another task involves the process of creating a round trip recommendation to the user. Furthermore, an overview about the base concept and workflow of the OTP graph building is described. Last but not least, this chapter also includes the methodology and the idea behind the GTFS transformation using realtime public transit data. More detailed information is given in the section 5 "Implementation".

4.1 Conceptual models for setting up the OTP

The conceptual models should help to understand the subject matter in a simple way. The use case model (Figure 12) reveals the different types of data for building a multimodal graph in OTP, whereas afterwards the desired route will be calculated using the route parameters of the user.

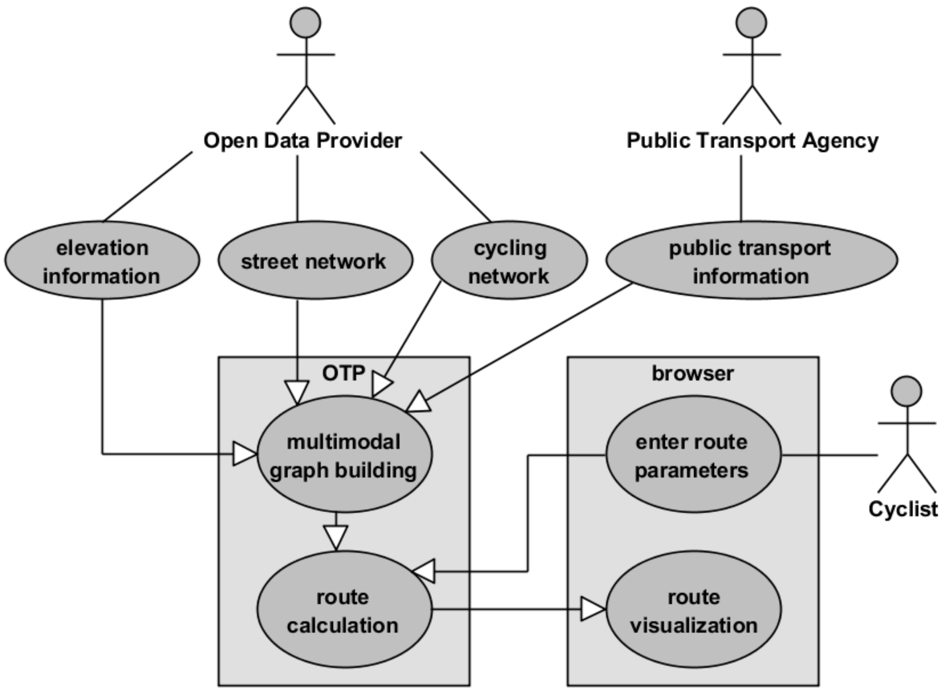

Fig. 12: Use case model of the workflow when using OTP

The data flow model (Figure 13) gives a more detailed overview of the system, and especially the transfer of data between the different actors. Out of this model, four different working tasks can be identified. The first task (1) deals with the conversion of time tables into the GTFS format that includes time-dependent information for trip planning. This has to be done if the public transport agency only provides raw time tables instead of a standardized GTFS feed (see 4.2). Another decision (2) that needs to be taken includes either the use of OSM or another data source such as GIP (Mandl-Mair 2014) for the creation of the multimodal graph (see 4.3). A further big issue deals with the integration of a round trip option for the user into OTP. Here, an algorithm needs to be implemented to recommend an attractive route to the cyclist, based on a desired time (3) and additional route parameters (4) like the type of bicycle or maximal slope (see 4.5).

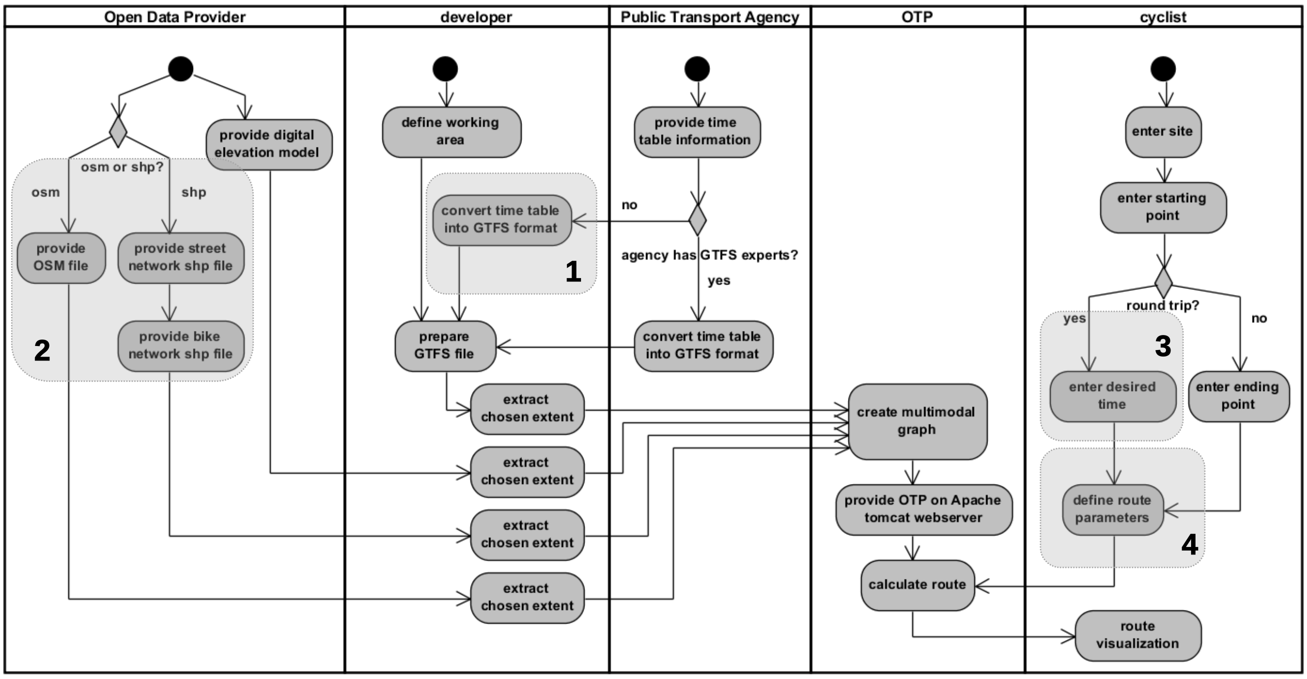

Fig. 13: Data Flow Model when using OTP

4.2 Transit data

An essential part of this research is the availability of transit data in the GTFS format to be able to consider time table information. In case that this information is not given in the desired format, the data needs to be prepared using available schedule information.
Each GTFS feed consists of a set of unique csv-text files (Figure 14), which, as already stated in section 3.1.4, are compressed in a ZIP package. All the required files have to contain exactly the previous described column headings to provide the information in a standardized way, by the means of a traditional database.

> *"The overall structure of the database tries to avoid duplicate information by creating cascading relationships from the most disaggregated information in the stop_times table to the most aggregated information in the agency table. As an example, a row of data in the stop_times table refers to the scheduled arrival and departure of a transit vehicle on a specific trip; that trip is categorized by a route which is categorized by the agency providing it."* (Wong 2013).

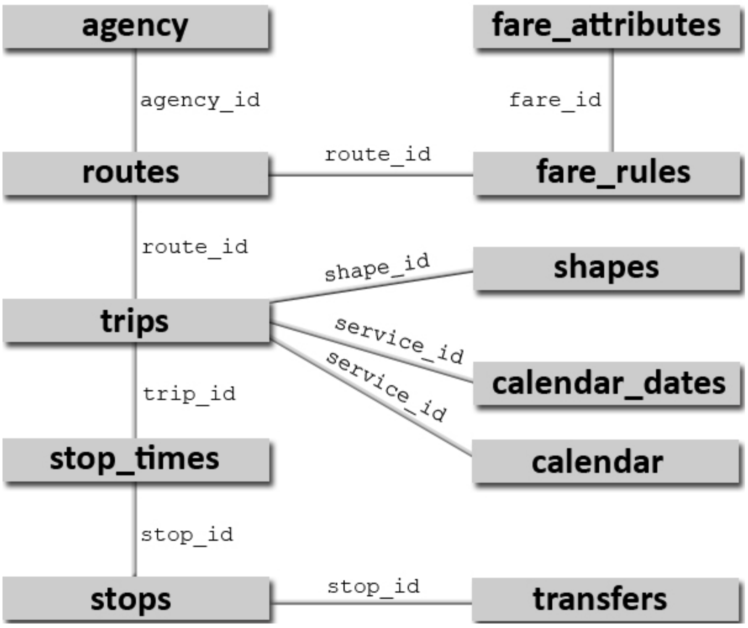

Fig. 14: General Transit Feed Specification (GTFS) relations

In Figure 15, the data view of a particular GTFS feed from Berlin is visible. Five of the formatted csv files are shown on the left side, as well as a list of all files that are located in the GTFS feed on the lower right side.

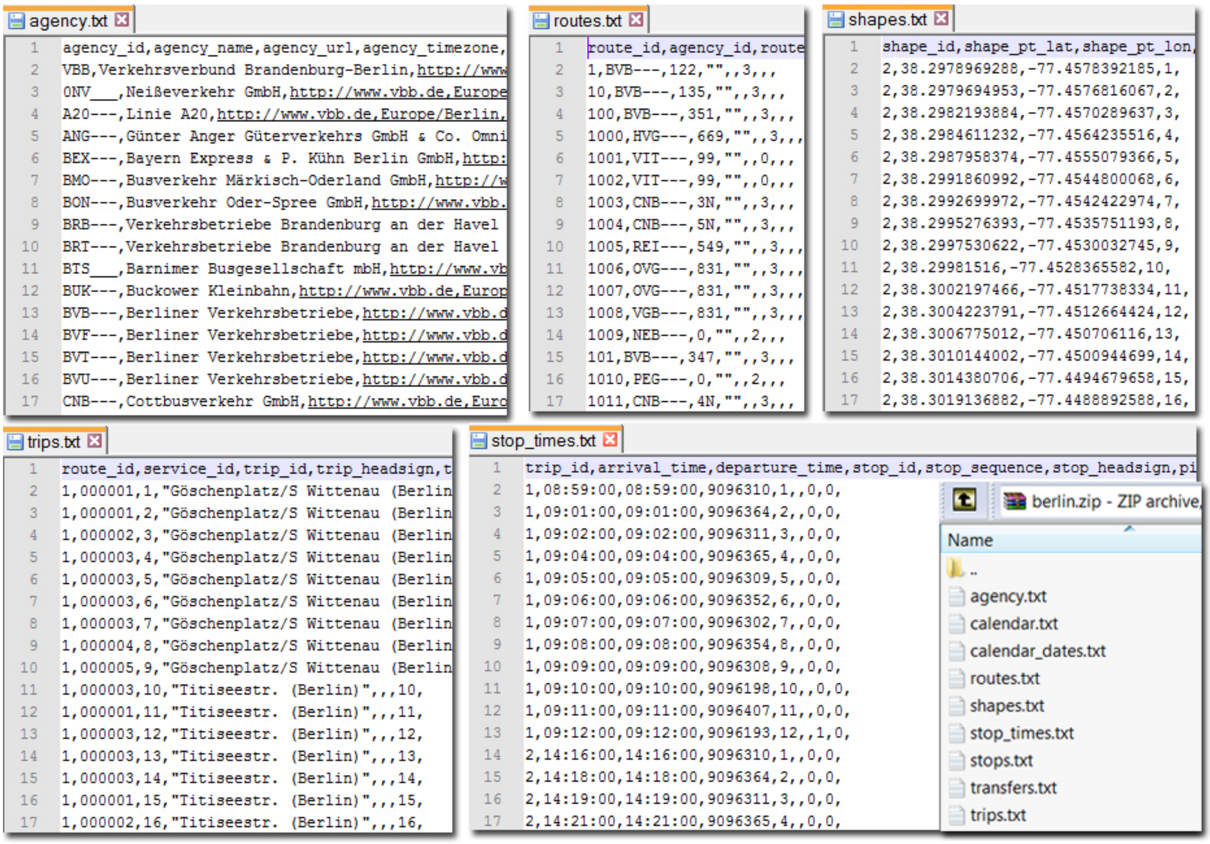

Fig. 15: Formatted General Transit Feed Specification (GTFS) files

There are several solutions available for converting time table information into the GTFS format[17]. Depending on the used schedule software like e.g. Trapeze FX[18], GIRO HASTUS[19], Trillium Solutions[20], Next Inside Transportation Software[21] and others, an export option to directly write GTFS files is provided.

In case the agency doesn't have commercial packages to directly export the GTFS feed, a small tool called "XLS Tools" (Heitzman 2014) was developed to support small agencies. Using this tool, the transit data can be manually entered into a pre-formatted Microsoft Excel sheet and afterwards being exported as a complete GTFS feed.

If transit data is available in a certain standard other than GTFS, existing tools can also convert these data into the GTFS feed format. An example is the GoogleTransitDataFeed[22], which is able to handle transit data in the TransXChange standard format for this purpose, or the "onebusaway-gtfs-realtime-from-siri-cli"[23] application that was designed to convert data in the SIRI format into the GTFS-realtime format.

In general, it is very difficult to obtain data from transit agencies that don't follow an Open Data (OD) approach (STK 2012) and is definitely not part of this research.

[17] http://code.google.com/p/googletransitdatafeed/wiki/OtherGTFSTools last access 04.02.2014
[18] http://www.trapezegroup.com/ last access 04.02.2014
[19] http://www.giro.ca/en/ last access 04.02.2014
[20] http://www.trilliumtransit.com/ last access 04.02.2014
[21] http://nextinsight.com/services.php last access 04.02.2014
[22] http://code.google.com/p/googletransitdatafeed/wiki/GoogleTransitDataFeed last access 04.02.2014
[23] http://developer.onebusaway.org/modules/onebusaway-gtfs-realtime-from-siri-cli/current/ last access 04.02.2014

4.3 OpenTripPlanner (OTP)

OTP offers an API for the development of applications as well as a web interface that enables users to plan a route under consideration of bicycle, pedestrian, transit and car attributes.

> *"It follows a client-server model with several map-based web interfaces included and exposes a REST API that can be used by third-party applications. OTP is fully multi-modal in that access to transit and transfers can occur on foot or by bicycle accounting for one-way streets, bicycle lanes, or the hilliness of terrain. OTP relies on open data standards including GTFS for transit and OpenStreetMap for street networks. In the new version of OTP currently under development, GTFS-Realtime updates are applied during the itinerary optimization process so late or cancelled vehicles are taken into account." (OTP 2013)*

As already described in Weyrer (2013), the OTP consists of the "Graph Building", the "Routing Engine" and the "Web Interface", which are the main components of the framework. The "Graph Building" component creates an object called "Graph.obj", which is the basis for the routing. The settings for that process like data source paths and restrictions are included in a configurations file. The "Routing Engine" uses this graph to calculate a route after a request from the "Web Interface". In case the route is computed, the route, the map and additional information like route instructions will be visualized in the "Web Interface".

4.3.1 Graph structure

In OpenTripPlanner, all the street and transport networks are represented as directed graphs. In Figure 16, a small extract of New York City with street and public transit information is represented. In the version of OTP ("master branch"), the model has shifted from a pattern-based graph into an edge-based graph (see Figure 17). However, the new model is built upon it. Because of the directed graph, the street segments consist of two edges, whereas each edge indicates one direction. This allows a limitation of different travel modes on one street (e.g. pedestrians in both directions, cars in one-way streets). Each transit stop is linked to a street vertex and has at least one node that is connected to a transit trip. Multiple nodes may be connected to a single transit stop, if several platforms exist, or if the station entrances are far away from each other. In this case, the street edge of the transit trip would be split. The transit trips will be described from the Board, Alight, Dwell and Hop edges. If trips have the same sequence of stops, they can be grouped to time-dependent PatternBoard, PatternAlight, PatternDwell and PatternHop edges. Time-dependent means that in this case their weight functions vary according to the time at which they are traversed. For example, a PatternBoard edge will look at the next departure time for its corresponding trip pattern and hence vary its weight. In Figure 16, two different trips from different subways, which have the same stops, are displayed (GitHub GraphStructure 2014).

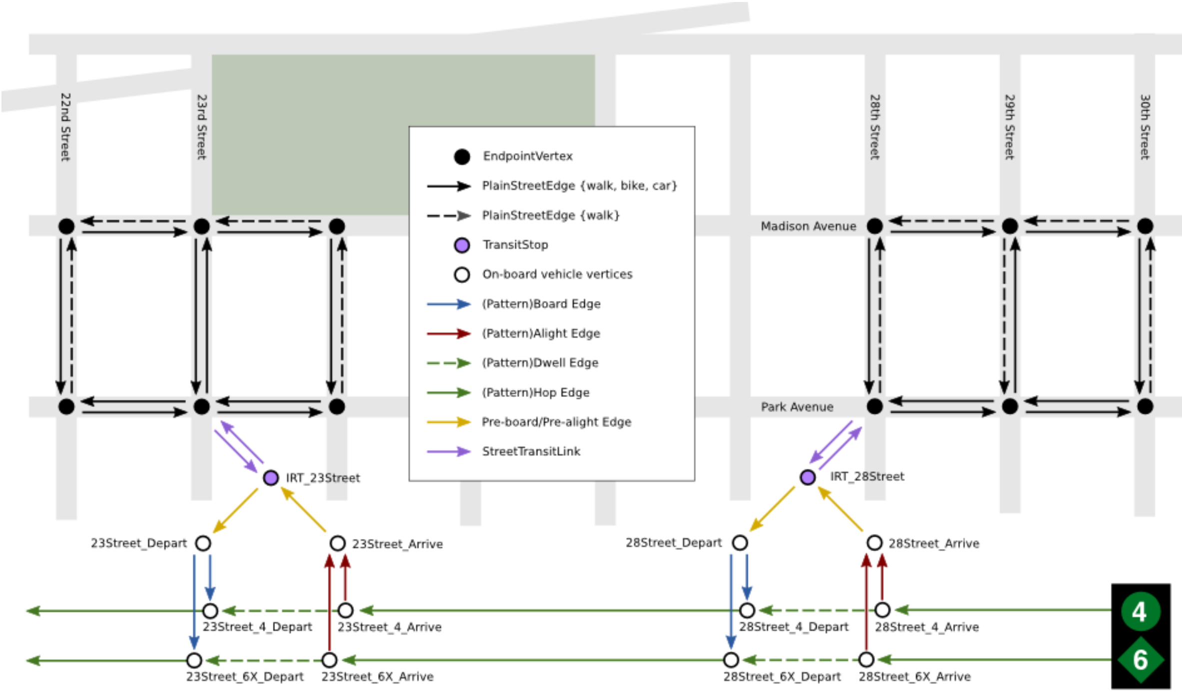

Fig. 16: Simple representation of the pattern-based Graph Structure in OTP

To enable an advanced performance like turn restrictions, the pattern-based representation from Figure 16 was changed to the edge-based representation in Figure 17. In order to do this, each vertex of the original graph is replaced by an edge and vice versa. In this view, a vertex means being on a street segment, whereas being on an edge means changing the street segment. Here, it is possible to apply turning restrictions to individual turns as well as different travel modes. The only thing that needs to be considered is that the computation of the shortest path can also loop itself (GitHub GraphStructure 2014).

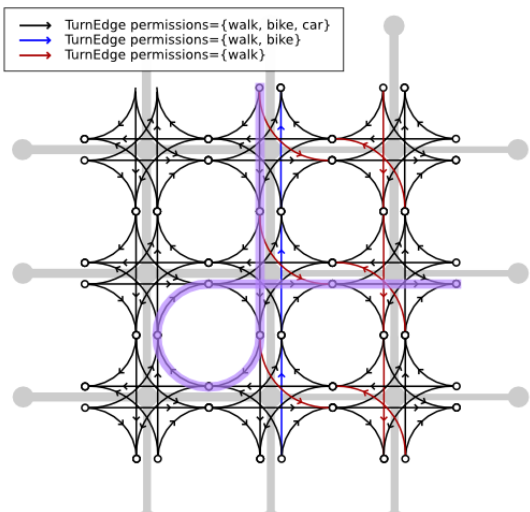

Fig. 17: Recent edge-based representation of the Graph Structure

4.3.2 Graph creation using shapefile or OSM

The input data source for constructing a street network can either be one or multiple shapefiles, or OpenStreetMap (OSM). In order to define travel permissions on different street segments, there are two main differences between these types of data. It is to mention that the following settings are only configurable this way in the “stable” branch of OTP. As described in section 5, the more common “master” branch will be used for this research, which doesn’t handle the configurations in XML files anymore, so this chapter is only for explanation purposes.

When using shapefiles as an input data source, a schema has to be defined that explains travel permissions of different street segments. Besides, an ID and name column has to be linked to the schema. The “PermissionConverter” deals with the task how the traffic is handled along both directions of a given street segment. It is possible to define default permission as well as user-defined ones. In Figure 18, the values ‘W’ and ‘A’ both define one-way streets, whereas the first element in the pair refers to travel along the street in one direction and the second one refers backwards (Github Graphbuilder 2014).

```xml
<bean id="shapefileStreetBuilder" class="org.opentripplanner.graph_builder.impl.shapefile.ShapefileStreetGraphBuilderImpl">
    <property name="featureSourceFactory">
        <bean class="org.opentripplanner.graph_builder.impl.shapefile.ShapefileFeatureSourceFactoryImpl">
            <property name="path" value="path/to/nyc/streets/streets.shp" />
        </bean>
    </property>
    <property name="schema">
        <bean class="org.opentripplanner.graph_builder.impl.shapefile.ShapefileStreetSchema">
            <property name="idAttribute" value="StreetCode" />
            <property name="nameAttribute" value="Street" />
            <property name="permissionConverter">
                <bean class="org.opentripplanner.graph_builder.impl.shapefile.CaseBasedTraversalPermissionConverter">
                    <property name="attributeName" value="TrafDir" />
                    <property name="defaultPermission" value="PEDESTRIAN_AND_BICYCLE" />
                    <property name="permissions">
                        <map>
                            <entry key="W" value="ALL,PEDESTRIAN" />
                            <entry key="A" value="PEDESTRIAN,ALL" />
                            <entry key="T" value="ALL,ALL" />
                        </map>
                    </property>
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

Fig. 18: XML based representation of the shapefile “PermissionConverter” in the configurations file (Github Graphbuilder 2014)

The use of OSM is handled in a different way than shapefiles. The “data provider” can either be a single or multiple .osm files, or an online provider that enables you to define a specific region that is going to be downloaded, using the public OSM REST API. As each street segment in OSM has certain tags associated with it, it is possible to obtain street names and permissions.
In the configuration, the “wayProperties” can consist out of multiple definitions to set restrictions on different types of roads. In Figure 19, ways with the tags

‘highway=motorway’ are only allowed to be used by cars. Bicycles are only allowed to drive on ways where a tag has the value ‘highway=motorway;cycleway=lane’ and are also respected to be less safe, so a safety value is defined for both directions. The third “WayPropertyPicker” indicates a mix of safety features - if a surface consists of gravel, it is considered to be twice as dangerous as a street segment without gravel. So, if a way is specified by the tags ‘highway=motorway’, ‘cycleway=lane’ and ‘surface=gravel’ the “safetyFeature” would be multiplied (1.5x2=3). According to this configuration, all permission properties can be set manually (Github GraphBuilder 2014).

```
<property name="wayPropertySet">
    <bean class="org.opentripplanner.graph_builder.impl.osm.WayPropertySet">
        <property name="wayProperties">
            <list>
                <bean class="org.opentripplanner.graph_builder.impl.osm.WayPropertyPicker">
                    <property name="specifier" value="highway=motorway" />
                    <property name="properties">
                        <bean class="org.opentripplanner.graph_builder.impl.osm.WayProperties">
                            <property name="permission" value="CAR" />
                        </bean>
                    </property>
                </bean>

                <bean class="org.opentripplanner.graph_builder.impl.osm.WayPropertyPicker">
                    <property name="specifier" value="highway=motorway;cycleway=lane" />
                    <property name="properties">
                        <bean class="org.opentripplanner.graph_builder.impl.osm.WayProperties">
                            <property name="safetyFeatures" value="1.5,1.5" />
                            <property name="permission" value="BICYCLE_AND_CAR" />
                        </bean>
                    </property>
                </bean>


                <bean class="org.opentripplanner.graph_builder.impl.osm.WayPropertyPicker">
                    <property name="specifier" value="surface=gravel" />
                    <property name="properties">
                        <bean class="org.opentripplanner.graph_builder.impl.osm.WayProperties">
                            <property name="safetyFeatures" value="2.0,2.0" />
                            <property name="safetyMixin" value="true" />
                        </bean>
                    </property>
                </bean>

                ...
            </list>
        </property>
        ...
    </bean>
</property>
```

Fig. 19: XML based representation of the OSM “WayPropertyPicker” in the configurations file (Github Graphbuilder 2014)

4.4 Round trip recommendation

A great function for tourist bicycle trip planners could be a round trip recommendation, where the user enters an origin and an amount of time he wants to schedule. For this, the routing engine has to include a process that calculates an appropriate tour with the desired length. This is also provided by the platform Komoot (Komoot 2013). The user defines the ride of bike he is going to drive, a starting point and an amount of time in hours. As a result, one up to several recommendations appear on the screen (Figure 20). For each tour, additional information like overall height difference, elevation profile and the ground composition is displayed.

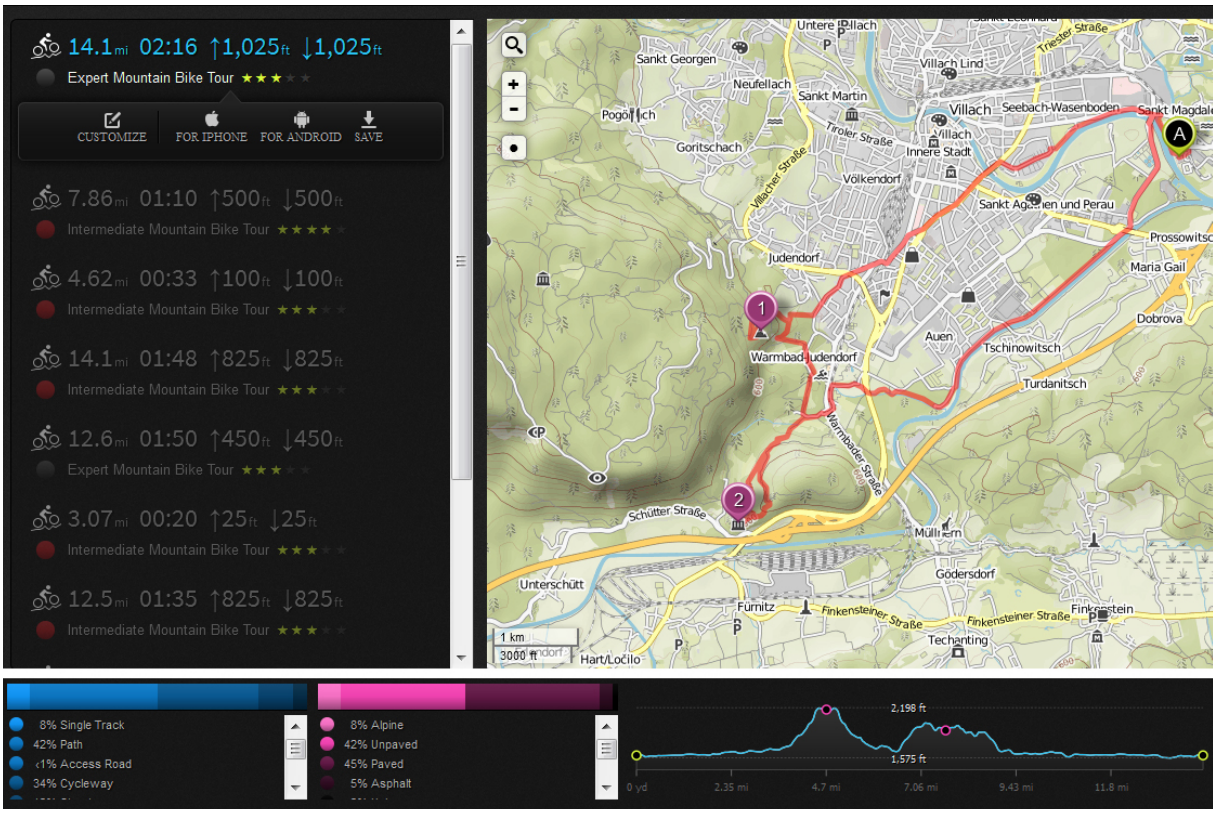

Fig. 20: Screenshot of a round trip created using the Komoot (2013) interface

When trying to extend the functionality of OTP by a round trip recommendation, a new algorithm has to be implemented. Zipf et al. (2006) already realized a similar approach for the navigation of pedestrians in a town. To do this, Points of Interests (POIs) have to be embedded as the main focus of attention. Especially in towns it is beneficial to use POIs for guided tourist round tours. Joest & Stille (2002) also concerned with this subject matter, using a generalization of the Travelling Salesman Problem (TSP), called Enhanced Profitable Tour Problem (EPTP). They implemented an algorithm that uses initial parameters like the start location, the total amount of time the visitor is willing to spend on the tour and the speed the person is travelling. The Integer Linear Program (ILP) is an NP-hard optimization problem which means that an optimal solution doesn't exist. It is designed to use the following objective function and constraints (Joest & Stille 2002):

maximize

$$\sum_{i \in V} p_i y_i + \sum_{i \in V} \sum_{j \in V} p_{ij} x_{ij} \quad (1)$$

subject to

$$\sum_{i \in V \setminus \{i\}} x_{ij} y_i = 0, \quad \forall\, i \in V, \quad (2)$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} y_i = 0, \quad \forall\, j \in V, \quad (3)$$

$$y_1 = 1, \quad (4)$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{i\}} t_{ij} x_{ij} + \sum_{i \in V} t_i y_i \le t_{max}\,, \quad (5)$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \ge y_h\,,$$

$$\forall\, S \subset V : 1 \in S, h \in V \setminus S, \quad (6)$$

$$x_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in A, \quad (7)$$

$$y_i \in \{0,1\}, \quad \forall\, i \in V. \quad (8)$$

G = (V,A)
G… weighted digraph
V… set of nodes y, V={1,...,n}
A… set of arcs x
every arc (i,j) ∈ A has price p_i and time t_i

(1) obj. function: maximize the price of nodes and arcs in the circle
(2),(3) every node in the circle must be connected with exactly one in bound and exactly one outbound arc
(4) start node must be part of the circle
(5) total time of the circle must be less or equal than t_{max}
(6) every node in the circle must be connected to the start node. Using this constraint together with constraints (2) und (3) en sures that there is only one circle
(7),(8) guarantee x_{ij} and y_i to be binary

The above mentioned method still works fine for creating round tours by means of POIs. In rural areas outside of a city, where it is considered to plan cycling tours, in many cases only a few POIs exist and so it is not the best way to follow this approach. A reference point for this matter is described in Stroemer (2006). It is stated that a tour suggestion in those areas should be performed using the geometry of line segments from the routing graph (Figure 21). To do this, a circular area that has the desired route length (if time is preferred, then you only have to know the speed of the movement) will be created, whereas the starting point needs to be intersected with the border of the circle. The alignment of the circle could either be done manually by the user, which requires additional user input. On the other hand, this could also be done in an automatic way by orienting the circle towards the line segment, starting from the origin.

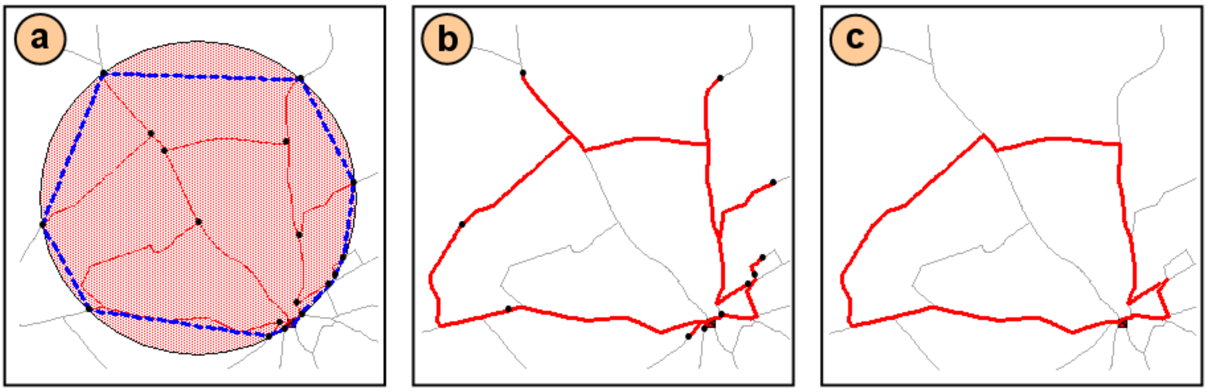

Fig. 21: Proceeding for the calculation of a round trip recommendation in rural areas (a) line segments in the circular buffer as well as convex hull (dotted line) of all coordinates intersecting the buffer (b) calculated route (c) route after deleting the segments which were passed twice (Stroemer 2006)

After this step, all points from the line segments that intersect the buffer will be selected and their coordinates are going to be identified. Then, the convex hull of all determined points will be computed to minimize the to-be processed point cloud. Stroemer (2006) revealed that the perimeter of the convex hull is supposed to have 70 up to 80 % of the desired route length to come to a satisfied result. If not, additional points can be added or deleted after this evaluation. A further step deals with the calculation of the round trip by means of generated points. Last but not least, each line segment in the route that needs to be travelled twice has to be removed in order to get a correct round trip.

In order to follow this approach in OTP, a detailed research of the available lines of code has to be performed to be able to work with objects inside. Particular methods also need to be extended or newly developed for both the computation and the visualization of the round trip recommendation in the web interface. As this kind of implementation would have exceeded the available research time, it was just mentioned theoretically.

4.5 Evaluating impacts of real-time data

A large part of this research deals with the analysis of real-time data in public transportation systems and their impact to the user of a trip planning system. Since described in the previous section, the OpenTripPlanner (OTP) is the chosen software for the analysis and is capable for all further measures to be done.

4.5.1 Real-time types and structure

As described in Section 3.1.5, the GTFS realtime specification consists of the three different feed types "trip updates", "service alerts" and "vehicle positions", which will be published as protocol buffers. Google Developers (2014a) describe protocol buffers as a flexible, efficient, automated mechanism for serializing structured data. It is extremely powerful when it comes to large datasets that are continuously changing in real-time. The information is encoded to a binary format, which makes it easy to read and write information in different programming languages.
Data providers even have the possibility to update or extend their data structure without changing already deployed programs that are compiled in the former structure. Compared to the widely used Extended Markup Language (XML) format, protocol buffers are much simpler, 3 to 10 times smaller and 20 up to 100 times faster. The big advantage for programmers is that the information can be accessed via classes (Google Developers 2014a). In the following Figure 22, a comparison of data size and throughput between the XML and the protocol buffer structure is represented. The chart shows three different object sizes, whereas a significant difference in the XML representation can be determined with an increasing amount of data. Concurrently, the throughput of XML decreases dramatically, whereas the throughput of protocol buffer messages just falls slightly. One interesting point in this illustration is that the throughput for objects up to 5kb remains almost the same for both XML and protocol buffer files.

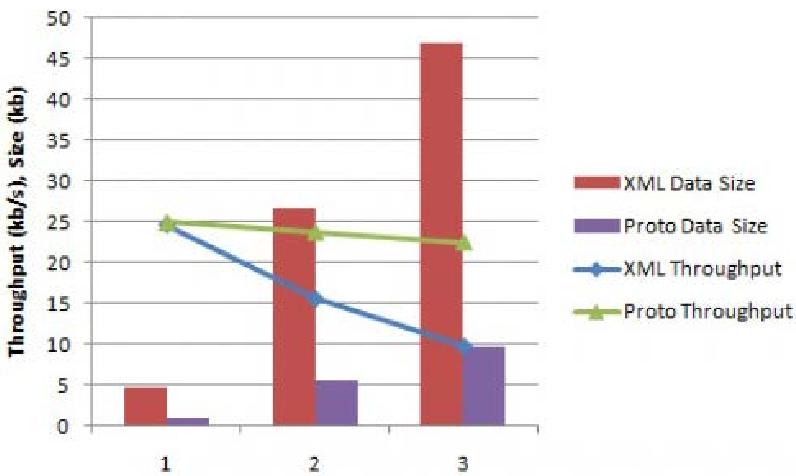

Fig. 22: Comparison of Data Size (bar graph) and Throughput (line graph) between XML and protocol buffer structure (Afroz Ahmad 2014)

The following Figure 23 shows a decoded extract of each of the three different protocol buffer files, visualized in three different colours (red, blue and green):

```
trip {
  trip_id: "10108181"
  start_date: "20140425"
  schedule_relationship: SCHEDULED
  1003: "\n\fVTN:22:22047"
}
stop_time_update {
  stop_sequence: 1
  arrival {
    delay: -185
    time: 1398440035
  }
  departure {
    delay: 0
    time: 1398440220
  }
  stop_id: "28033"
}
stop_time_update {
  stop_sequence: 2
  arrival {
    delay: 112
    time: 1398440392
  }
  departure {
    delay: 187
    time: 1398440467
  }
  stop_id: "27908"
}
stop_time_update {
  stop_sequence: 3
  arrival {
    delay: 205
    time: 1398440605
  }
  departure {
    delay: 237
    time: 1398440637
  }
  stop_id: "27787"
}
stop_time_update {
  stop_sequence: 4
  arrival {
    delay: 215
    time: 1398440735
  }
  departure {
    delay: 222
    time: 1398440742
  }
  stop_id: "27786"
}
stop_time_update {
  stop_sequence: 5
  arrival {
    delay: 198
    time: 1398440778
  }
  departure {
    delay: 218
    time: 1398440798
  }
  stop_id: "27897"
}
stop_time_update {
  stop_sequence: 6
  arrival {
    delay: 187
    time: 1398440827
  }
  departure {
    delay: 269
    time: 1398440909
  }
  stop_id: "27828"
}
stop_time_update {
  stop_sequence: 7
  arrival {
    delay: 233
    time: 1398440933
  }
  departure {
    delay: 272
    time: 1398440972
  }
  stop_id: "27831"
}
stop_time_update {
  stop_sequence: 8
  arrival {
    delay: 241
    time: 1398441001
  }
  departure {
    delay: 253
    time: 1398441013
  }
  stop_id: "28068"
}
```

```
trip {
  trip_id: "10543436"
  start_date: "20140414"
  schedule_relationship: SCHEDULED
  1003: "\n\017QBUZZ:d003:1049"
}
position {
  latitude: 52.74404
  longitude: 6.880617
}
current_stop_sequence: 25
current_status: IN_TRANSIT_TO
timestamp: 1397492208
stop_id: "118120"
vehicle {
  id: "QBUZZ:3003"
  label: "3003"
  1003: "\b\001\022\024Mercedes-Benz Citaro"
}
1003: "\b0"
--------------------------------------------
trip {
  trip_id: "8813930"
  start_date: "20140414"
  schedule_relationship: SCHEDULED
  1003: "\n\fCXX:M300:249"
}
position {
  latitude: 52.323486
  longitude: 4.786151
}
current_stop_sequence: 18
current_status: STOPPED_AT
timestamp: 1397492256
stop_id: "45938"
vehicle {
  id: "CXX:9224"
  label: "9224"
}
1003: "\b\205\004"
```

```
informed_entity {
  stop_id: "47262"
}
cause: HOLIDAY
effect: UNKNOWN_EFFECT
header_text {
  translation {
    text: "ivm de jaarmarkt route lijn 14 gestr advies halte station "
    language: "nl"
  }
}
description_text {
  translation {
    text: "Oorzaak : Jaarmarkt \nivm de jaarmarkt route lijn 14 gestr advies halte station \n"
    language: "nl"
  }
}
active_period {
  start: 1398151800
  end: 1398873600
}
```


Source of GTFS realtime feeds: http://gtfs.ovapi.nl/new/
Protocol buffer files were decoded to plain text

TripUpdates: 04/25/2014 12:32:00pm (EDT)
VehiclePositions: 04/14/2014 12:19:03pm (EDT)
ServiceAlerts: 05/01/2014 12:24:50pm (EDT)

Fig. 23: Three different decoded protocol buffer types

4.5.2 GTFS transformation approach

Protocol buffers are compressed and encoded very efficiently and need to be decoded when it comes to analysis purposes. The definition of the data structure is included in a ".proto"-file, which can be used as a general reference to create objects in different programming languages. The compiler for this process is Google's "protoc.exe", which is a command line tool that generates the code for any given protocol definition in the languages C++, Java or Python[24].
The idea behind this part of research is to adapt static timetable information according to available realtime data. As GTFS realtime is an extension to the standard GTFS format, both information can be easily combined. There are two different ways of how the transformation of the original (static) timetable could be achieved:

1. Using TripUpdate messages:

   TripUpdate messages contain all deviation information about realtime-capable trips. For each trip that is represented in the file, predicted arrival and/or departure times for stops along that route are given. If there is no trip update available for a particular scheduled trip, it will be inferred that no realtime data is available. Otherwise, the delay information states that the trip is not running on time. The picture in the previous page shows that for each trip in the realtime feed, one StopTimeUpdate exists for all associated stops. Depending on the current position of the vehicle, these updates imply past or future stop times. Each StopTimeUpdate contains the absolute time and/or a delay for each arrival and departure linked to that specific stop (Google Developers 2014c). The predictions are calculated using a route guidance system that permanently checks the current position and the amount of time to reach the future stops in that particular scheduled trip. The arrival time is the time where the vehicle opens the door and the departure time when it closes the door (Transit Developers 2014a). Each trip is ordered by stop sequences, which means that in the absence of trip update information for certain stops will change all subsequent stops. An example from Google Developers (2014c) would be:

   For a particular trip with 30 stops, three StopTimeUpdates are provided:
   - *delay of 200 seconds for stop_sequence 5*
   - *delay of 100 seconds for stop_sequence 20*
   - *delay of 50 seconds for stop_sequence 25*

   The information will be interpreted as:
   - *delay of 200 seconds for all stop_sequences between 5 and 20*
   - *delay of 100 seconds for all stop_sequences between 20 and 25*
   - *delay of 50 seconds for all stop_sequences between 25 and 30*

[24] https://code.google.com/p/protobuf/downloads/detail?name=protoc-2.5.0-win32.zip&can=2&q= last access 20.05.2014

The TripUpdate messages change their content continuously, which means that as soon as an update is received, it will be encoded and stored in the update file. The information of all available StopTimeUpdates could be used to update the arrival and departure times of the original timetable. The limitation of this approach is that a calculated trip using this type of realtime information is only valid for exactly that timestamp where the trip was computed. Because of this constraint, another approach to modify the static time deals with collecting real-time VehiclePositions.

2. Using VehiclePositions messages:

   VehiclePositions are used to provide location information about single vehicles in the whole public transit network. Each entity that is available in the message is supposed to contain a trip descriptor that identifies the trip the vehicle is riding on, as well as a vehicle descriptor which specifies the physical entity itself. Additional information about whether the vehicle has stopped or is in transit to a specific stop sequence allows a more detailed view on the behaviour (Google Developers 2014d).
   This type of realtime message could be used to achieve knowledge about real positions of all vehicles that offer this information. Collecting this data over a certain time period could provide exact timestamps when a single vehicle arrived and departed at sequenced stops. As a result, instead of predicted times from TripUpdate messages, true timestamps can be used to modify a timetable 'as it would have been optimal' for the collected time period. The original GTFS feed could serve to create a change set.

4.5.3 Workflow for the evaluation of real-time information

The following section describes the essential steps of realtime data collection, the workflow for the modification of the GTFS feed, the communication with the OTP REST interface and the analysis of the stored realtime data. Another main part deals with the visualisation of calculated routes and delay information. The first step deals with the collection of real-time data from the Open Public Transit Data Provider. The collected feeds need to be decoded and stored in a central spatial database. Additionally, the original GTFS feed needs to be imported there as well. Using the recorded real-time positions, it is possible to create a modified GTFS feed that includes collected timestamps. The OpenTripPlanner (OTP) is able to handle both the original and modified GTFS feed, with and without tripUpdate delay information, to analyze differences in route times. Furthermore, both the predicted and observed delays can be compared to obtain divergences as well. In chapter 4.5.4, five different scenarios are defined, which are supposed to show differences in travelling times of different passenger types. This is where the REST API of the OTP is going to be used.
Figure 24 visualizes the whole workflow for the real-time analysis concept:

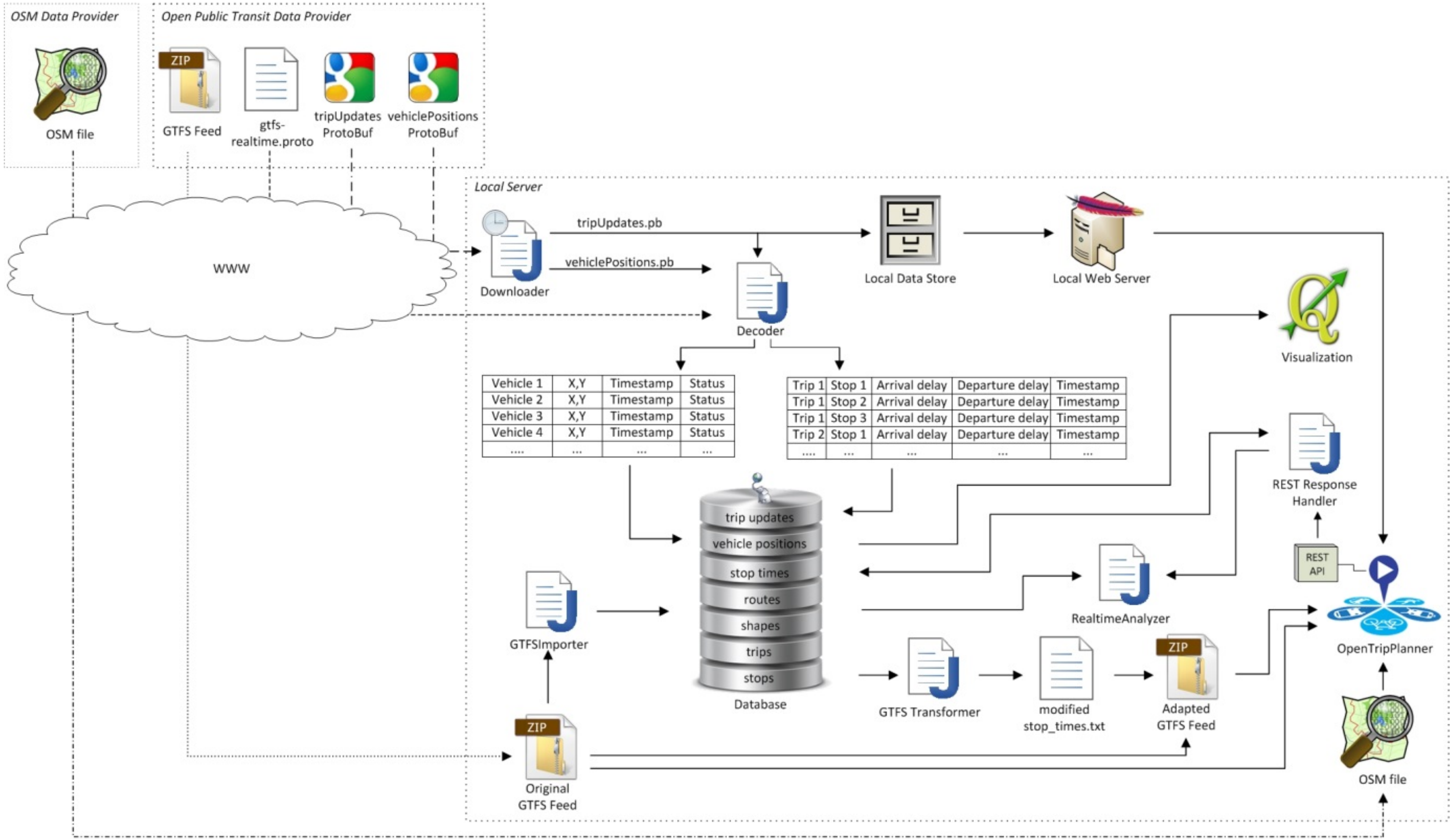

Fig. 24: Workflow for evaluating the effectiveness of real-time public transit information

4.5.4 Definition and concept of scenarios

For this research, five different scenarios were defined to “simulate” the trip planning and travelling activity of a single passenger that wants to get from one place A to another place B. The REST API of the OpenTripPlanner is able to handle route requests from origin to destination, as well as additional parameters like the route mode, maximal walk distance, time information and date. The response is an XML structure that can be evaluated. The five scenarios should simulate different manners of passengers that plan a route, either via an online web page or by phone. They also differ in using available tripUpdate delay estimates or not.

1) print trip on paper, not considering tripUpdates:

   In the first scenario, a user calculates a trip using the OTP without tripUpdates. Because of this, no delay information will be considered during the computation. It is assumed that the user prints the route and attempts to follow exactly the written instructions since he isn’t aware of that area. If he changes to the next bus line and misses the planned one, he has to wait for the following one that departs for the same route like on the plan.

2) print trip on paper, considering tripUpdates:

   This scenario follows the same behaviour like the first one, except the case that the initial calculation considers tripUpdates. Because of this, future delay estimates are integrated during the route calculation phase, which could have impacts in the travelling period.

3) Smartphone without realtime route computation:

   In the third scenario, it is assumed that the passenger has a smartphone and uses it for the initial route calculation to get from the defined origin to the destination. He follows the route on all the bus lines displayed on the screen, until he misses one because of unexpected delays. As soon as this circumstance happens, the passenger uses his smartphone again to calculate a further route from the current position to the destination. That process is done every time until he reaches his target.

4) Smartphone including realtime route computation:

   In this case, a “realtime simulation” of the fastest route is supposed to be calculated. Like a car navigation system that continuously considers traffic jams, accidents and other situations to visualize the fastest route, it is assumed to follow the instance using tripUpdates. The idea is to permanently calculate the fastest route and inform the user in case of a new computed route, where he has to change buses. As the trip planer would calculate a walking route to the next stop while the bus is on tour, the continuous re-calculation is supposed to be performed at each stop. That process is done until the passenger reaches the desired place.

5) optimal route calculation using modified GTFS feed:

   The last scenario handles the “optimal” case of a route that could have been used by the passenger. The routing graph contains timetable information from the modified GTFS feed and is only true for routes that lie in the past. As the modified GTFS feed contains the observed arrival and departure times that were collected during the real-time data collection phase, the calculated route in this scenario is supposed to be the “optimal” one the passenger could have taken.

The route request of all different scenarios looks the same; the only difference in each of the scenarios is the used routing graph, the use of tripUpdates and if requests will be renewed recursively in the case of a missed bus. The request is sent to the port of the local running OTP instance and includes the optional parameters like route mode, optimization criteria and maximal walking distance. Additionally, the coordinates of the origin and destination will be appended as well as the date and desired time. The response offers an XML structure where all the details of the routes are included (Figure 25). This framework includes request parameters, reverse geocoded origin and destination, and itineraries. Itineraries consist of start and end time of the route as well as legs that represent either a walking or a bus route. Each bus route (=leg) gives information about the tripID, the stopID of that trip where the path starts and ends. In the case of used tripUpdates during route calculation, each leg also gives information if the selected trip is affected by real-time or not (Figure 26).

REST request

http://localhost:8094/otp-rest-servlet/plan?mode=BUSISH%2CWALK&optimize=QUICK&maxWalkDistance=2500&toPlace=51.9172721622353%2C5.84924237909843&fromPlace=52.197716056277%2C6.3664405916241&date=2014-05-28&time=09:00:00

REST response


```
- <response>
  + <requestParameters></requestParameters>
  + <debugOutput></debugOutput>
  - <plan>
      <date>2014-05-28T03:00:00-04:00</date>
    - <from orig="">
        <name>Postelstraat</name>
        <lon>6.3648094571486045</lon>
        <lat>52.19784636323023</lat>
      - <geometry>
          {"type": "Point", "coordinates": [6.3648094571486045,52.19784636323023]}
        </geometry>
      </from>
    - <to orig="">
        <name>service road</name>
        <lon>5.848524</lon>
        <lat>51.9175095</lat>
      - <geometry>
          {"type": "Point", "coordinates": [5.848524,51.9175095]}
        </geometry>
      </to>
    - <itineraries>
      - <itinerary>
          <duration>12085.0</duration>
          <startTime>2014-05-28T09:53:27+02:00</startTime>
          <endTime>2014-05-28T13:14:52+02:00</endTime>
          <walkTime>899</walkTime>
          <transitTime>7770</transitTime>
          <waitingTime>3416</waitingTime>
          <walkDistance>1142.322567211175</walkDistance>
          <walkLimitExceeded>false</walkLimitExceeded>
          <elevationLost>0.0</elevationLost>
          <elevationGained>0.0</elevationGained>
          <transfers>3</transfers>
        - <legs>
          + <leg mode="WALK" route="" agencyTimeZoneOffset="7200000" interlineWithPreviousLeg="false" rentedBike="false"></leg>
          + <leg mode="BUS" route="54" agencyName="Arriva" agencyUrl="http://www.arriva.nl" agencyTimeZoneOffset="7200000" routeType="3" routeId="4"
            interlineWithPreviousLeg="false" tripShortName="1014" headsign="Zutphen Busstation" agencyId="ARR" tripId="11496856" serviceDate="20140528"
            routeShortName="54" routeLongName="Zutphen - Rijssen" rentedBike="false"></leg>
          + <leg mode="WALK" route="" agencyTimeZoneOffset="7200000" interlineWithPreviousLeg="false" rentedBike="false"></leg>
          + <leg mode="BUS" route="504" agencyName="Syntus" agencyUrl="http://www.syntus.nl" agencyTimeZoneOffset="7200000" routeType="3" routeId="648"
            interlineWithPreviousLeg="false" tripShortName="31126" headsign="Buurtbus Eerbeek" agencyId="SYNTUS" tripId="12494309" serviceDate="20140528"
            routeShortName="504" routeLongName="Buurtbus Zutphen - Buurtbus Eerbeek" rentedBike="false"></leg>
          + <leg mode="WALK" route="" agencyTimeZoneOffset="7200000" interlineWithPreviousLeg="false" rentedBike="false"></leg>
          + <leg mode="BUS" route="43" agencyName="Syntus" agencyUrl="http://www.syntus.nl" agencyTimeZoneOffset="7200000" routeType="3" routeId="623"
            interlineWithPreviousLeg="false" tripShortName="36997" headsign="Arnhem" agencyId="SYNTUS" tripId="12509511" serviceDate="20140528"
            routeShortName="43" routeLongName="Arnhem via Dieren - Apeldoorn" rentedBike="false"></leg>
          + <leg mode="WALK" route="" agencyTimeZoneOffset="7200000" interlineWithPreviousLeg="false" rentedBike="false"></leg>
          + <leg mode="BUS" route="331" agencyName="Breng" agencyUrl="http://www.breng.nl" agencyTimeZoneOffset="7200000" routeType="3" routeId="7187"
            interlineWithPreviousLeg="false" tripShortName="59" headsign="Nijmegen Weezenhof" agencyId="BRENG" tripId="9289688" serviceDate="20140528"
            routeShortName="331" routeLongName="Arnhem IJsseloord 2 - Nijmegen Weezenhof" rentedBike="false"></leg>
          + <leg mode="WALK" route="" agencyTimeZoneOffset="7200000" interlineWithPreviousLeg="false" rentedBike="false"></leg>
          </legs>
          <tooSloped>false</tooSloped>
        </itinerary>
      </itineraries>
    </plan>
  </response>
```


Fig. 25: Extract from an XML response after a sample REST request to the OTP

```
- <leg mode="BUS" route="43" agencyName="Syntus" agencyUrl="http://www.syntus.nl" agencyTimeZoneOffset="7200000" routeType="3" routeId="623"
  interlineWithPreviousLeg="false" tripShortName="36997" headsign="Arnhem" agencyId="SYNTUS" tripId="12509511" serviceDate="20140528"
  routeShortName="43" routeLongName="Arnhem via Dieren - Apeldoorn" rentedBike="false">
    <startTime>2014-05-28T11:43:00+02:00</startTime>
    <endTime>2014-05-28T12:46:10+02:00</endTime>
    <departureDelay>0</departureDelay>
    <arrivalDelay>0</arrivalDelay>
    <realTime>false</realTime>
    <distance>25717.407386373</distance>
    <pathway>false</pathway>
  - <from stopIndex="24" stopSequence="25">
      <name>Eerbeek, Centrum</name>
    - <stopId>
        <agencyId>ARR</agencyId>
        <id>21271</id>
      </stopId>
      <stopCode>40543007</stopCode>
      <lon>6.067213</lon>
      <lat>52.104738</lat>
      <arrival>2014-05-28T11:24:29+02:00</arrival>
      <departure>2014-05-28T11:43:00+02:00</departure>
    - <geometry>
        {"type": "Point", "coordinates": [6.067213,52.104738]}
      </geometry>
    </from>
  - <to stopIndex="69" stopSequence="70">
      <name>Arnhem, Rozet</name>
    - <stopId>
        <agencyId>ARR</agencyId>
        <id>163634</id>
      </stopId>
      <stopCode>40009261</stopCode>
      <lon>5.905298</lon>
      <lat>51.980399</lat>
      <arrival>2014-05-28T12:46:10+02:00</arrival>
      <departure>2014-05-28T12:46:10+02:00</departure>
    - <geometry>
        {"type": "Point", "coordinates": [5.905298,51.980399]}
      </geometry>
    </to>
  - <legGeometry>
      <length>497</length>
    + <points></points>
    </legGeometry>
    <steps/>
    <duration>3790.0</duration>
  </leg>
```


Fig. 26: Extract from a “BUS”-leg out of the XML response

In Java, the architecture for reading XML documents is called Java Architecture for XML Binding (JAXB) and defines an interface to write Java objects. It is very simple and can also be extended very easily. A tutorial is given in Vogella (2014).

The main entities of the response will be filtered, converted into Java objects and afterwards stored into the spatial database. This information then allows the comparison of trip times in different scenarios.

4.6 Study area and geodata

For this research, two different study areas were chosen: Austria and Netherlands. Public transit agencies in Austria are not willing to offer open public transit data in the GTFS format (DerStandard 2013, Kurier 2013), not to mention realtime data. At least a small test feed was offered for this research to depict a purpose.

4.6.1 Austria

In Figure 27, the distribution of the ÖBB railway stop points from the test GTFS feed is visualized. Most of the points are located in the Southern part of Austria (Carinthia), whereas some stops from single international trips are included in the feed, too. It is important to mention that this test feed only contains an extract out of all available railway routes in Austria.

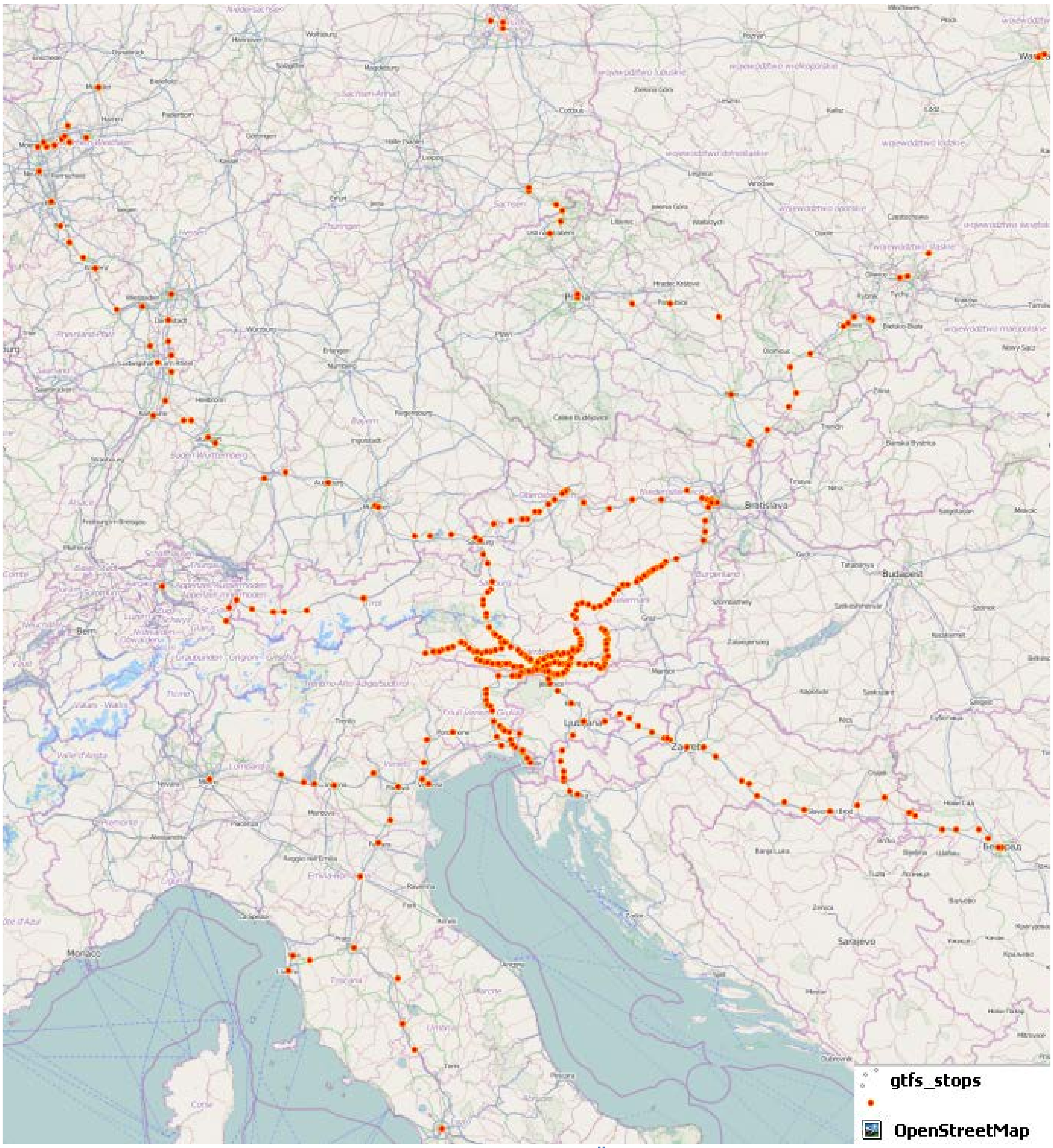

Fig. 27: Overview of railway stop points from the ÖBB GTFS test feed

4.6.2 Netherlands

The lack of available open public transportation data in Austria has forced to shift the study area to Netherlands, which is the only country in Europe where GTFS-realtime feeds are available as Open Data nowadays (Transitfeeds 2014). The shapefile of the stops, which are visible in Figure 27 and 28, were converted using a JQL query language code from DrJTS (2011) and were executed using the software JEQL (2014). Figure 28 gives an overview about all available stops of the GTFS feed from Netherlands.

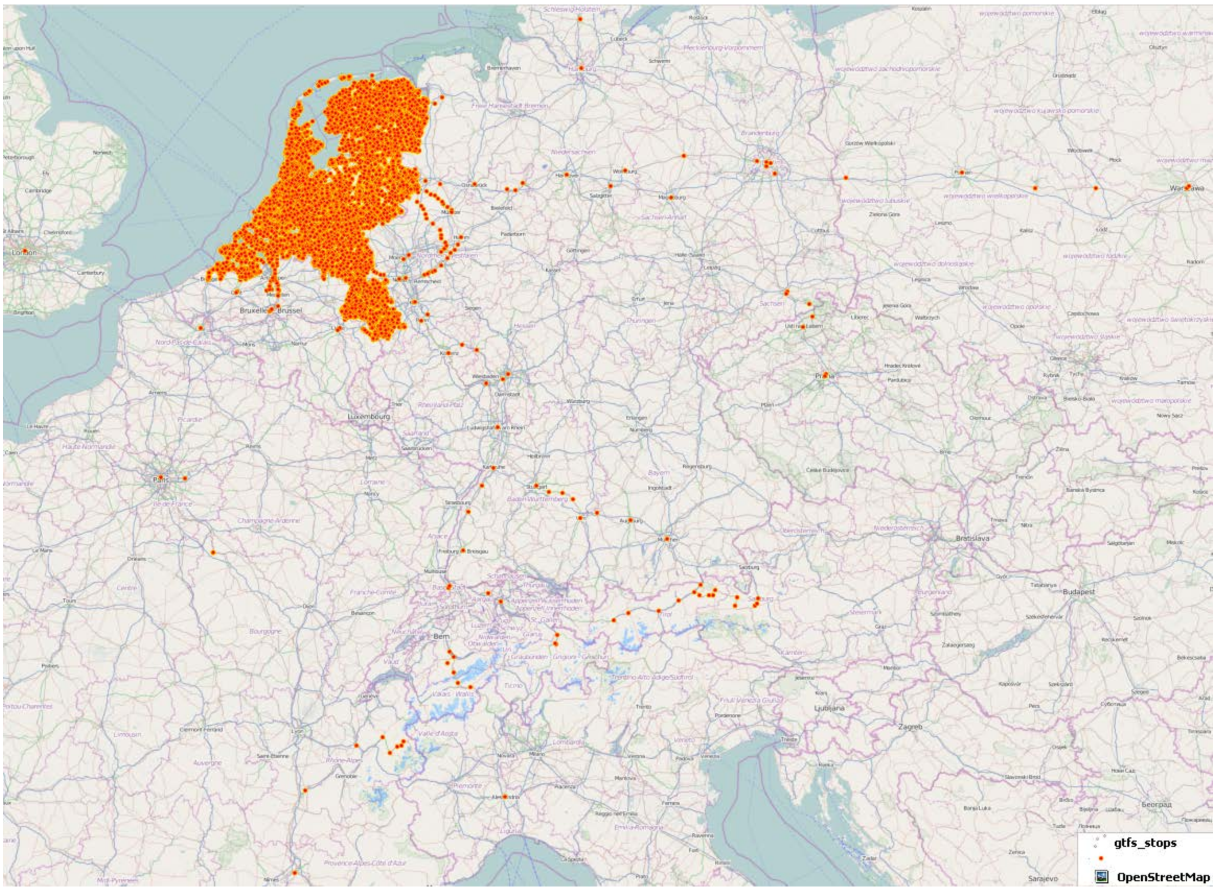

Fig. 28: Overview about the extent of all OVapi stops which are included in the GTFS feed

All necessary transit information is available on the web directory of OVapi (OVapi 2014), whereas the street network is downloaded as an OSM file from Geofabrik (Geofabrik NL 2014). Using this information, it is possible to create the routing graph in OTP.

As a data source for the elevation data, free digital elevation model (DEM) data is available at United States Geological Survey (USGS) Earth Explorer[25] or on DititalGlobe® ImageFinder[26]. For Carinthia (Austria), an Open Governmental Data (OGD) portal offers the elevation data in 1m resolution (Data KTN 2014).

---

[25] http://earthexplorer.usgs.gov/ last access 14.02.2014
[26] https://browse.digitalglobe.com/imagefinder/main.jsp last access 14.02.2014

5. Implementation

5.1 OpenTripPlanner

This section gives a detailed overview about the functionalities of the OpenTrip-Planner (OTP) (OTP 2013) as well as the implementation of the research project itself. The three main components “Graph Building”, “Routing Engine” and “Web Interface” are used during the entire research.
The OpenTripPlanner project is freely available on GitHub (GitHub 2014) and can either be downloaded using Linux or a GitHub extension for Windows console (GitHub TwoMinutes 2014), or using a development environment like Eclipse[27] or Netbeans[28] (GitHub Install 2014). GitHub not only provides a single instance of OTP, but several different branches that focus on different topics like bike safety, bike parking, cars, graph, OTP analyst etc. All these branches are further developments of older ones or just focus on a certain region (e.g. berlin-bike). It is also important to know that there are two different code structures available, one based on the “stable” branch, which has been developed until about autumn 2013[29], and the newer “master” branch that is the most recent one. The main difference between the “older” stable branch and the “newer” master branch is the structure of different modules, shown in Table 5:

Table 5: Comparison between the OTP “stable” and the “master” branch (Github OTP 2014a & 2014b)
*** … subprojects which have to be built separately**

STABLE	
opentripplanner-routing	Core routing algorithms, data-structures, and libraries
opentripplanner-api-webapp	Webapp providing a REST web api to the trip planning engine
opentripplanner-webapp	Webapp providing web-based user interface to trip planning engine
opentripplanner-graph-builder	Command line tool for configuring and building the trip planner graph
opentripplanner-gui	Graph visualizer primarily useful for development and troubleshooting
opentripplanner-integration	Integration tests
MASTER	
otp-core	core routing algorithms, data structures, libraries, and stand-alone server
otp-analyst-client	A Javascript client focusing on OTP analyst web service visualizations
otp-geocoder	A servlet that converts addresses to geographic locations using web services

[27] https://www.eclipse.org/downloads/ last access 21.02.2014
[28] https://netbeans.org/downloads/ last access 21.02.2014
[29] https://github.com/opentripplanner/OpenTripPlanner/tree/stable last access 21.02.2014

otp-leaflet-client	The newer Javascript client providing a map-based UI for trip planning
otp-openlayers-client	The original Javascript client providing a map-based UI for trip planning
otp-rest-servlet	A servlet that provides the OTP REST API within a servlet container
otp-thrift-api	A Thrift API supporting lover-level queries than the REST API.
otp-admin-client	A client for Administration
otp-datastore *	A Play-based backend for logging OTP queries
otp-gvsig *	An OpenTripPlanner-based extension to GVSIG

The whole documentation that can be found in the OTP wiki (GitHub Wiki 2014) is mainly based on the stable branch. Up to now (Feb 21, 2014), documentation for the master branch is very rare, so the only way to get information about the latest branch is to get into contact with the developers in different Google Groups (OTP Developers 2014, OTP Users 2014a, GTFSRT 2014, Transit Developers 2014b). According to a blog entry from an active OTP developer (Feb 21, 2014), a proper information for the upcoming version 1.0 is going to be prepared (OTP Users 2014b).

5.2 Bike & transit for Austria

The reason why this research also deals with GTFS data from Austria is to show the opportunities of open transit information for the public. The main goal of this part was to demonstrate the combination of bike and transit together with elevation information, which could be interesting for tourists that want to plan a bike tour all over the country. For this demonstration, the “older” OpenTripPlanner version was chosen because this version is capable to integrate elevation data coming from another source than National Elevation Data (NED) data in the United States (OTP Users 2014d). The web interface is also able to visualize the elevation profile during the trip (Figure 29). The OSM dataset from Austria is downloadable via Geofabrik (Geofabrik AUT 2014); Elevation data is available on the federal governmental portal of Carinthia (Data KTN 2014). The configuration for the OpenTripPlanner is attached in the appendix and describes how the three different data sources (GTFS feed, OSM file, elevation data) are applied to create the routing graph.

Fig. 29: Visualization of a route that uses a combination out of bike (grey) and transit (red). The location is central Carinthia, Austria.

Other than Google Maps, the OTP is capable to combine bike and transit in a single route calculation. This function offers a great opportunity for tourism-focused areas, e.g. a round-trip function as described in section 4.4. As the aims of this project have shifted from the round-trip approach to the evaluation of real-time public transit data, the former solution statement enables an approach for further development.

5.3 Evaluating the effectiveness of real-time public transit information in Netherlands

The main part of this research deals with the evaluation of real-time data in multimodal networks. Real-time data are available from the open public transit data provider, which is in Netherlands the company OVapi (2014). Besides, they also provide the digital timetable of the whole transit network as a GTFS feed. These files are accessible manually or via programming interfaces. For this work, Java (2014) is used as the main programming language. The development platform that is used is called “Eclipse”, which is comprised of extensible frameworks, tools and runtimes for building, deploying and managing executable software (Eclipse 2014).

5.3.1 Realtime data collection

Realtime data will be collected for a certain time range and a certain time interval that has been defined by the user. Both, the tripUpdates and the vehiclePositions messages need to be decoded in a way that they are readable as objects in the programming environment. As the compressed information is stored in the protocol buffer file, the easiest way to process it is to use classes that store the data in the memory. In order to do this, a Google compiler called “protoc.exe” [30] uses the description of a gtfs-realtime.proto document (described in section 3.1.5) to create classes for the used programming language. These classes contain methods that enable the encoding of data into a binary file or loading a binary file to decode it (Scriptol 2014).
The objects will be stored in the PostgreSQL database that uses a PostGIS extension for spatial queries. The tables have the following schema:

vehiclePositions:

`trip_id integer,`	Identifier the current trip
`curr_stop_seq smallint,`	current stop sequence of the trip
`curr_status text,`	current movement status of the vehicle
`stop_id integer,`	stop identifier
`timestmp timestamp with time zone,`	timestamp of downloaded message
`pos_geom geometry(Point,4326)`	position of the vehicle

tripUpdates:

`trip_id integer,`	Identifier of the current trip
`start_date date,`	start date of trip update
`schedule_relationship text,`	trip schedule status
`stop_sequence smallint,`	stop sequence affected from update
`arrival_delay integer,`	estimated arrival delay at stop
`arrival_time time without time zone,`	estimated arrival time at stop
`departure_delay integer,`	estimated departure delay at stop
`departure_time time without time zone,`	estimated departure time at stop
`stop_id integer,`	stop identifier
`stop_schedule_relationship text,`	stop schedule status
`timestmp timestamp with time zone`	timestamp of downloaded message

[30] https://code.google.com/p/protobuf/ last access 29.05.2014

5.3.2 GTFS import

Besides, transit timetable data need to be stored in the database, too. To accomplish this, a GTFS Importer reads the single CSV-files from the feed and saves them according to their data structure (see Figure 14) into the separate database tables. The central data store enables queries and changes to be performed, as well as the linkage of single tables using their unique identifiers (see Figure 15). A main advantage of importing the feeds into a database is the computation speed since no CSV-files need to be browsed any more.

5.3.3 GTFS transformation

Position data messages are collected over time for a certain time interval, e.g. every 10 seconds for 12 hours. This enables a precise evaluation of time and space of each vehicle and gives valuable information to create a modified GTFS feed that contains exact departure and arrival times from the collected time period for each specific trip. In Table 6, an extract of a particular trip is given, whereas each row represents the information out of one specific timestamp where the vehiclePositions message has been collected.

Table 6: Extract of a particular trip from the table vehicle_positions (*) column timestmp is interpreted as Eastern Time (GMT-4)

TRIP_ID	CURR_ STOP_SEQ	CURR_STATUS	STOP_ID	TIMESTMP (*)	ST_X	ST_Y
8451805	1	STOPPED_AT	0	05.28.2014 9:55:28	4,681904	51,815403
8451805	1	STOPPED_AT	0	05.28.2014 9:55:38	4,681904	51,815403
8451805	1	STOPPED_AT	0	05.28.2014 9:55:48	4,681904	51,815403
8451805	1	STOPPED_AT	0	05.28.2014 9:55:58	4,681904	51,815403
8451805	1	STOPPED_AT	0	05.28.2014 9:56:08	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:56:18	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:56:28	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:56:38	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:56:48	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:56:58	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:57:08	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:57:18	4,681904	51,815403
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:57:28	4,680538	51,81377
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:57:38	4,680538	51,81377
8451805	1	IN_TRANSIT_TO	91996	05.28.2014 9:57:48	4,680538	51,81377
8451805	3	IN_TRANSIT_TO	9382	05.28.2014 9:57:58	4,679016	51,81279
8451805	3	IN_TRANSIT_TO	9382	05.28.2014 9:58:08	4,679016	51,81279
8451805	3	IN_TRANSIT_TO	9382	05.28.2014 9:58:18	4,679016	51,81279
8451805	4	STOPPED_AT	0	05.28.2014 9:58:28	4,676542	51,81333
8451805	4	IN_TRANSIT_TO	9384	05.28.2014 9:58:38	4,676542	51,81333
8451805	4	IN_TRANSIT_TO	9384	05.28.2014 9:58:48	4,676542	51,81333
8451805	5	STOPPED_AT	0	05.28.2014 9:58:58	4,673791	51,81487
8451805	5	IN_TRANSIT_TO	9383	05.28.2014 9:59:08	4,673791	51,81487
8451805	5	IN_TRANSIT_TO	9383	05.28.2014 9:59:18	4,673791	51,81487
8451805	6	STOPPED_AT	0	05.28.2014 9:59:28	4,671066	51,81569
8451805	6	STOPPED_AT	0	05.28.2014 9:59:38	4,671066	51,81569

8451805	6	STOPPED_AT	0	05.28.2014 9:59:48	4,671066	51,81569
8451805	6	STOPPED_AT	0	05.28.2014 9:59:58	4,671066	51,81569
8451805	6	STOPPED_AT	0	05.28.2014 10:00:08	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:00:18	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:00:28	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:00:38	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:00:48	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:00:58	4,671066	51,81569
8451805	6	IN_TRANSIT_TO	9381	05.28.2014 10:01:08	4,671066	51,81569
8451805	7	STOPPED_AT	0	05.28.2014 10:01:18	4,668056	51,81708
8451805	7	IN_TRANSIT_TO	9236	05.28.2014 10:01:28	4,668056	51,81708
...	...	...	...	...	...	...

In the data transformation phase, the aim is to filter information out of each trip collected. The current status specifies if a bus has “stopped at” or is “in transit to” a certain stop. The time range in where a bus had the status “stopped at” a particular stop needs to be recorded for the modification of the original GTFS feed. Furthermore, the first timestamp will be used for the observed arrival time and the last timestamp will be used for the observed departure time. For instance, the original GTFS feed had an arrival and departure time of “15:59:00” at stop nr. 6 (Table 7), whereas the observed arrival time was at “15:59:28” and the observed departure time was at “10:00:08”. It is important to know that the timestamp in the database was recorded in Eastern Time (GMT-4), which is 6 hours behind the Central European Time (GMT+2).

Table 7: Extract of a particular trip from the table stop_times

TRIP_ID	STOP_ID	ARRIVAL_TIME	DEPARTURE_TIME	STOP_SEQUENCE
8451805	91996	"15:54:00"	"15:54:00"	1
8451805	9382	"15:56:00"	"15:56:00"	3
8451805	9384	"15:57:00"	"15:57:00"	4
8451805	9383	"15:58:00"	"15:58:00"	5
8451805	9381	"15:59:00"	"15:59:00"	6
8451805	9236	"16:01:00"	"16:01:00"	7
8451805	9307	"16:02:00"	"16:02:00"	9
8451805	9358	"16:02:00"	"16:02:00"	10
8451805	9190	"16:03:00"	"16:03:00"	11
8451805	9298	"16:04:00"	"16:04:00"	12
8451805	9239	"16:04:00"	"16:04:00"	13
8451805	3429	"16:05:00"	"16:05:00"	14
8451805	4331	"16:07:00"	"16:07:00"	16
8451805	3901	"16:10:00"	"16:10:00"	17

The modified arrival and departure times need to be packed into the original GTFS feed to be able to calculate routes with these times. As the stop_times.txt from the original feed was directly imported into the database, the change set may be exported into a stop_times.txt again. This file may replace the original one to result in a modified (adapted) GTFS feed.

5.3.4 OTP and REST API

In order to analyze the effectiveness, behaviour and quality of available realtime information, the OpenTripPlanner (OTP) provides a Representational State Transfer (REST) – Application Programming Interface (API) that returns the route request as an Extendex Markup Language (XML) response with all the route details. As described in chapter 4, the OTP needs a GTFS feed as input for the public transportation network and timetables as well as an OSM source as routing network for walking or bicycle distances. While working with the realtime data for this research, only walking will be considered in the multimodal route planning. To include tripUpdate information during the route planning process, the OTP is also capable of integrating the protocol buffer from a web server source. In other words, during the initial configuration process before starting the OTP instance, a “graph.properties”-file needs to store information about the tripUpdate source that is defined in the url parameter:

```
rt.type = stop-time-updater
rt.sourceType = gtfs-http
rt.defaultAgencyId = ARR
rt.url = http://localhost/tripUpdates/tripUpdates.pb
```

5.3.5 Realtime analysis

In the realtime analysis stage, the stored and calculated information will be evaluated. The first part deals with the description of observed travel time delays, whereas the effect of delay during the day is considered. For this analysis, the delay information of each single bus stop can be highlighted for every hour during the day. Besides, the percentage of realtime affected stops will be determined. For this, it will be distinguished between stops that were part of the vehiclePositions feeds and those that were part of tripUpdates feeds. Diagrams visualize the information in a human readable format. Besides, random routes will be calculated for all scenarios and travelling time differences between these scenarios will be compared.

5.3.6 Visualization

Every route that is calculated in any scenario will be stored in the PostGIS database, including spatial information. QuantumGIS is capable of handling PostGIS tables that can be visualized for the creation of maps. One part deals with the creation of maps including delay information and a comparison of tripUpdate and vehiclePosition densities, the other one focusses on the visualization of created trips from different scenarios.

6. Results

For this research, the whole process of data collection and analysis has been accomplished on May 28th 2014, from 7:00 am until 8:00 pm. On this day, vehiclePositions were gathered every 10 seconds, whereas tripUpdates have been downloaded every 30 seconds. The reason for the high collection frequency for vehiclePositions is the desired accuracy in the estimation of arrival and departure times. The GTFS feed as well as the OSM file from Netherlands was gathered from the same day as well. The following sections give an overview about the analysis of the recorded information.

6.1 Descriptive statistics of collected data

The descriptive statistics in Table 8 gives information about the total amount of collected data. The GTFS feed that was used for the import of public transit information into the database contains 2.383 routes that will be described by 713.199 trips and 72.256 stops. More than 4.5 million shapes form 9.236 shapelines to be visualized on a map. Almost 16 million individual stop_times give arrival and departure time information at every single stop of each trip. Furthermore, in the 13 hours of data collection, about 258.9 million tripUpdate entries and 16.9 million vehiclePositions were gathered. The amount of different trips in all tripUpdate entries is 73.947, whereas in the tripUpdate feed that is used for the analysis in scenario 2, about 10.500 trips are involved. Using all recorded vehiclePositions, an amount of 72.612 trips was comprised, which is about 10% of all available trips. During the creation process of the modified stop_times using vehiclePositions, a total amount of 52.296 complete trips were written into the new timetable that is used in the modified graph. This means that the timetable information of 7.3% of all available trips was changed in that process.

Table 8: Total amount of collected data

GTFS feed	
routes	2.383
trips	713.199
shapes	4.665.512
shapelines	9.236
stop_times	15.962.597
stops	72.256
GTFS real-time feeds	
tripUpdate entries	258.916.245
amount of trips in tripUpdates	73.947
trips in tripUpdate feed from '2014-05-28 03:00:00-04'	10.451
vehiclePositions	16.963.413
amount of trips in vehiclePositions	72.612
trips with same trip_id in vehiclePositions and tripUpdates	70.144
creation of modified stop_times	
trips without gaps	72.010
trips complete	52.296

6.2 Interpolation maps of real-time information

Each tripUpdate contains information about trips and hence all the past and future stops on these trips that are affected by delay information. Because of this, it is possible to say how often every single stop was affected by tripUpdates. For each stop, there is a fixed amount of trips running through it, so the relative amount of tripUpdates affected stops can be computed. The following map (Figure 30) illustrates an interpolation (kriging) map, where the mentioned percentage information of stops is used to create a contour surface with higher and lower tripUpdates density.

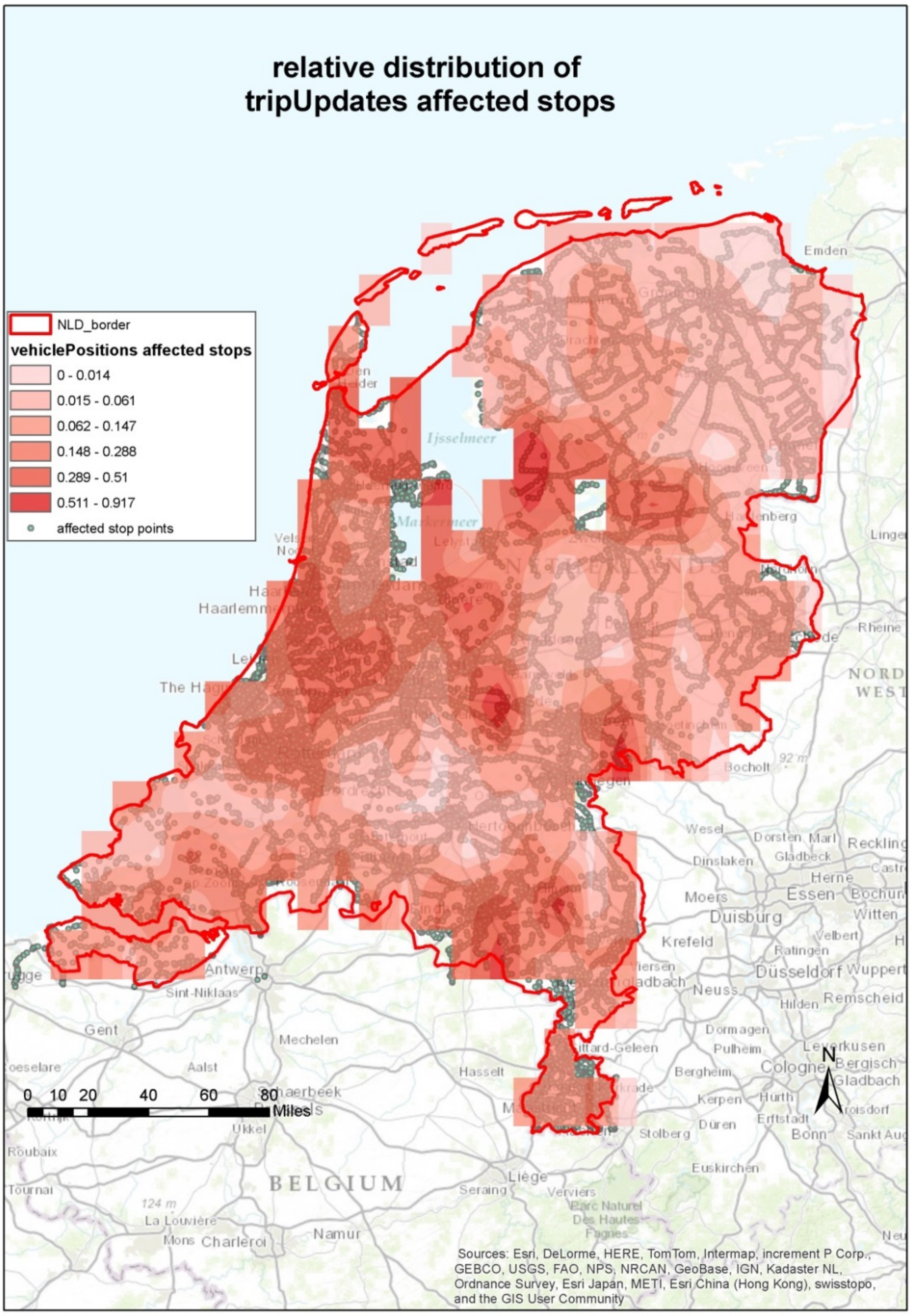

Fig. 30: Interpolation map of tripUpdates percentage at stops

Similar to tripUpdates, vehiclePositions give information about the specific trip a vehicle has been driving. Since it may be possible that the time where a vehiclePositions feed was collected was during the trip, not all stops were affected by this type of realtime information. Nevertheless, it is possible to calculate the amount of stops affected by vehiclePositions and calculate the relative amount regarding the total amount of trips through that stop. The following interpolation map (Figure 31) visualizes vehiclePositions density at single stops.

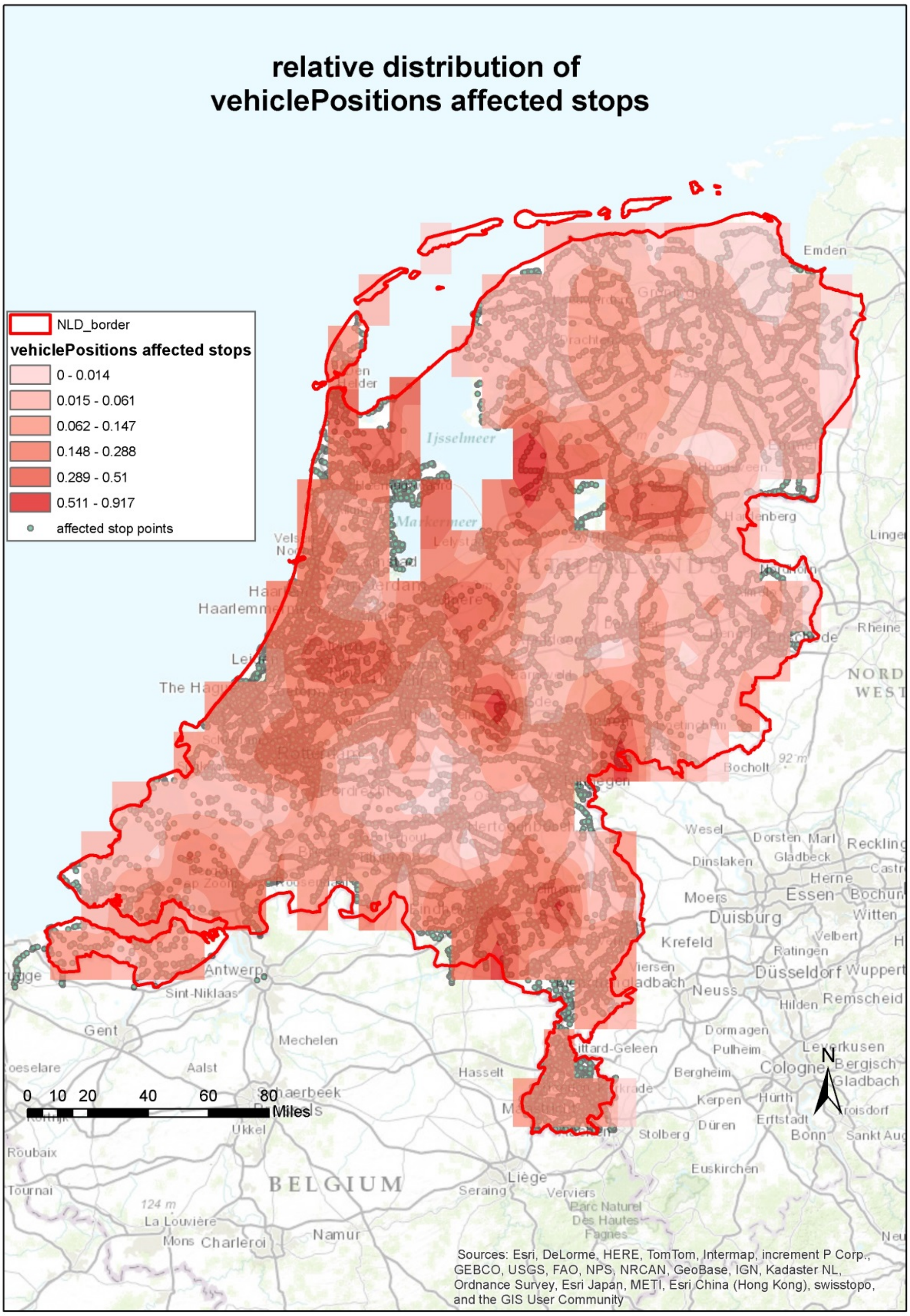

Fig. 31: Interpolation map of vehiclePositions percentage at stops

For the creation of the difference map (Figure 32), the relative amount of tripUpdates at every single stop was subtracted from the relative amount of vehiclePositions at every single stop. It is obvious that in the central area of Netherlands, especially in the Eastern of Amsterdam, the amount of tripUpdates is higher than the amount of vehiclePositions, whereas the bigger part of the country is considered with equal distribution.

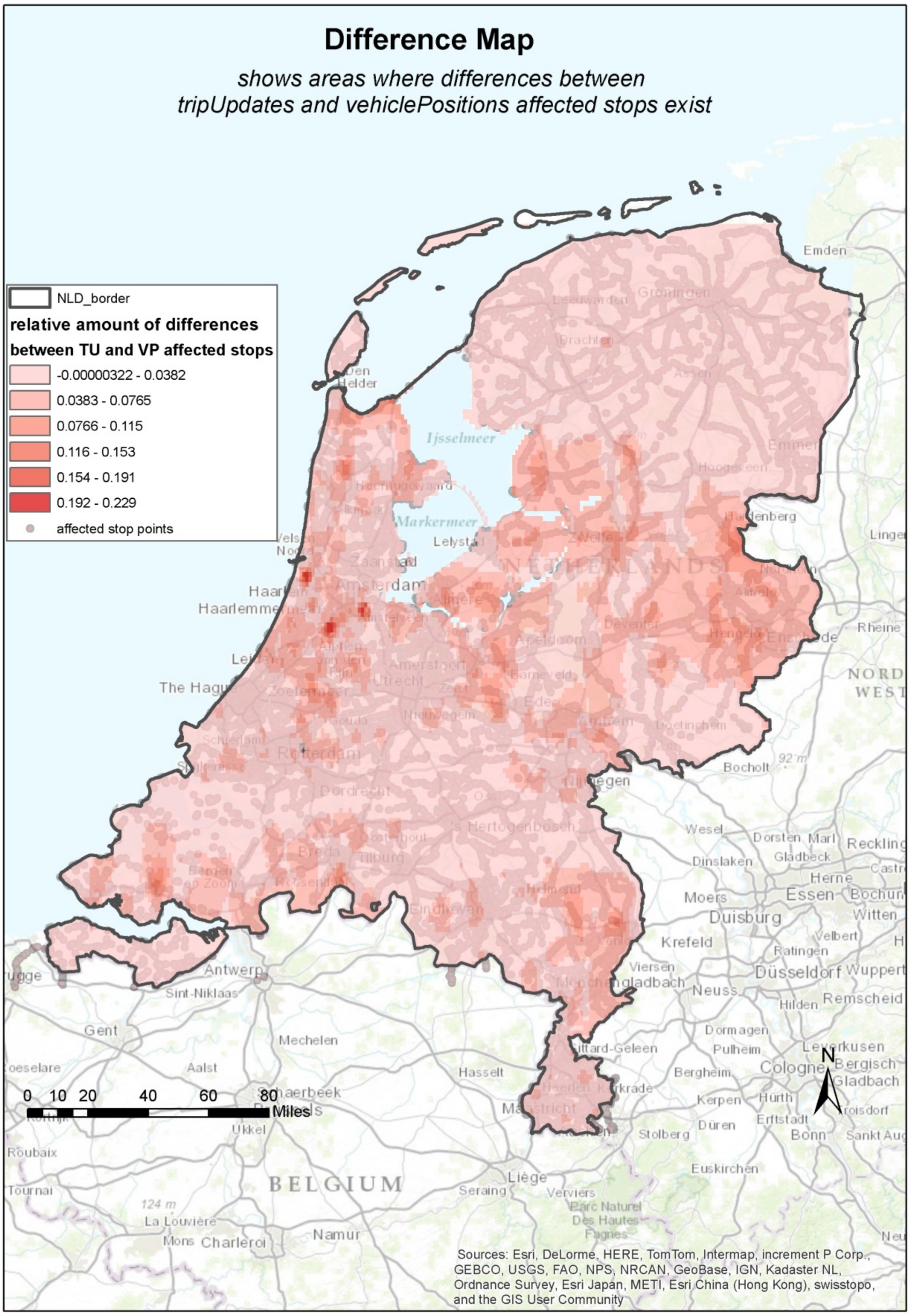

Fig. 32: Difference Map of both interpolation maps

6.3 Impact of vehiclePositions

As described in section 5.3.2, the modification of the arrival and departure times due to gathered vehiclePositions only works if these feeds will be recorded over a long time period. The more vehicles publish their positions, the more accurate and complete is the modified timetable. Furthermore, the higher the collection frequency, the more accurate are arrival and departure times. The effect of this type of realtime data can be observed in the section 6.2, where the scenario 5 makes use of the modified stop times in the GTFS feed. Table 9 shows an example of a modified timetable from a single trip:

Table 9: Extract of a particular trip from the table stop_times_modified (compared to Table 7)

TRIP_ID	STOP_ID	ARRIVAL_TIME	DEPARTURE_TIME	STOP_SEQUENCE
8451805	91996	"15:55:28"	"15:56:08"	1
8451805	9382	"15:57:58"	"15:57:58"	3
8451805	9384	"15:58:28"	"15:58:28"	4
8451805	9383	"15:58:58"	"15:58:58"	5
8451805	9381	"15:59:28"	"16:00:08"	6
8451805	9236	"16:01:18"	"16:01:18"	7
8451805	9307	"16:02:28"	"16:02:38"	9
8451805	9358	"16:02:48"	"16:03:08"	10
8451805	9190	"16:03:28"	"16:03:28"	11
8451805	9298	"16:04:28"	"16:04:28"	12
8451805	9239	"16:04:58"	"16:05:08"	13
8451805	3429	"16:06:38"	"16:06:48"	14
8451805	4331	"16:07:48"	"16:08:38"	16
8451805	3901	"16:10:58"	"16:10:58"	17

Due to data quality issues and cases where the vehicle didn't stop at several stations, the recorded vehiclePositions feed hasn't always included the exact sequence of a particular trip. If there was a stop missing or there was no information about "STOPPED AT" a current stop, arrival and departure times for these stops had to be estimated to complete the modified timetable. For this, all stops that were missing were interpolated to the nearest recorded vehicle position of that specific trip. For all the stops that have a sequential status "IN TRANSIT TO" a particular stop, the first timestamp of the nearest position to this stop was taken as arrival and departure time. In Figure 33, an extract of recorded vehiclePositions for a selected trip is shown on the left side. After matching with the original timetable from that trip, arrival and departure times were adapted to the recorded timestamps and saved as the modified timetable of that trip.

	trip_id integer	curr_stop_seq smallint	curr_status text	stop_id integer	timestmp timestamp with time zone	pos_geom geometry(Point,4326)	st_x double precision	st_y double precision
247	8775680	9	IN TRANSIT	49681	2014-05-28 12:58:12-04	0101000020E6100000	4.856905	52.301895
248	8775680	9	IN TRANSIT	49681	2014-05-28 12:58:22-04	0101000020E6100000	4.856905	52.301895
249	8775680	13	STOPPED AT	0	2014-05-28 12:58:32-04	0101000020E6100000	4.858561	52.301895
250	8775680	13	STOPPED AT	0	2014-05-28 12:58:42-04	0101000020E6100000	4.858561	52.301895
251	8775680	13	IN TRANSIT	32887	2014-05-28 12:58:52-04	0101000020E6100000	4.858561	52.301895
252	8775680	13	IN TRANSIT	32887	2014-05-28 12:59:02-04	0101000020E6100000	4.858561	52.301895
253	8775680	13	IN TRANSIT	32887	2014-05-28 12:59:12-04	0101000020E6100000	4.858561	52.301895
254	8775680	13	IN TRANSIT	32887	2014-05-28 12:59:22-04	0101000020E6100000	4.858561	52.301895
255	8775680	14	IN TRANSIT	36200	2014-05-28 12:59:32-04	0101000020E6100000	4.863348	52.301323
256	8775680	14	IN TRANSIT	36200	2014-05-28 12:59:42-04	0101000020E6100000	4.863348	52.301323
257	8775680	14	IN TRANSIT	36200	2014-05-28 12:59:52-04	0101000020E6100000	4.863348	52.301323
258	8775680	14	IN TRANSIT	36200	2014-05-28 13:00:02-04	0101000020E6100000	4.863348	52.301323
259	8775680	14	IN TRANSIT	36200	2014-05-28 13:00:12-04	0101000020E6100000	4.863348	52.301323
260	8775680	14	IN TRANSIT	36200	2014-05-28 13:00:22-04	0101000020E6100000	4.863348	52.301323
261	8775680	15	IN TRANSIT	46529	2014-05-28 13:00:32-04	0101000020E6100000	4.868515	52.302048
262	8775680	15	IN TRANSIT	46529	2014-05-28 13:00:42-04	0101000020E6100000	4.868515	52.302048
263	8775680	15	IN TRANSIT	46529	2014-05-28 13:00:52-04	0101000020E6100000	4.868515	52.302048
264	8775680	15	IN TRANSIT	46529	2014-05-28 13:01:02-04	0101000020E6100000	4.868515	52.302048
265	8775680	16	STOPPED AT	0	2014-05-28 13:01:12-04	0101000020E6100000	4.87308	52.302795
266	8775680	16	STOPPED AT	0	2014-05-28 13:01:22-04	0101000020E6100000	4.87308	52.302795
267	8775680	16	STOPPED AT	0	2014-05-28 13:01:32-04	0101000020E6100000	4.87308	52.302795
268	8775680	16	IN TRANSIT	42461	2014-05-28 13:01:42-04	0101000020E6100000	4.87308	52.302795
269	8775680	16	IN TRANSIT	42461	2014-05-28 13:01:52-04	0101000020E6100000	4.87308	52.302795
270	8775680	16	IN TRANSIT	42461	2014-05-28 13:02:02-04	0101000020E6100000	4.87308	52.302795
271	8775680	16	IN TRANSIT	42461	2014-05-28 13:02:12-04	0101000020E6100000	4.87308	52.302795
272	8775680	17	STOPPED AT	0	2014-05-28 13:02:22-04	0101000020E6100000	4.876168	52.301983
273	8775680	17	STOPPED AT	0	2014-05-28 13:02:32-04	0101000020E6100000	4.876168	52.301983
274	8775680	17	STOPPED AT	0	2014-05-28 13:02:42-04	0101000020E6100000	4.876168	52.301983
275	8775680	17	IN TRANSIT	34078	2014-05-28 13:02:52-04	0101000020E6100000	4.876168	52.301983
276	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:02-04	0101000020E6100000	4.876168	52.301983
277	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:12-04	0101000020E6100000	4.876168	52.301983
278	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:22-04	0101000020E6100000	4.876168	52.301983
279	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:32-04	0101000020E6100000	4.876168	52.301983
280	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:42-04	0101000020E6100000	4.876168	52.301983
281	8775680	17	IN TRANSIT	34078	2014-05-28 13:03:52-04	0101000020E6100000	4.8858943	52.300198
282	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:02-04	0101000020E6100000	4.8858943	52.300198
283	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:12-04	0101000020E6100000	4.8858943	52.300198
284	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:22-04	0101000020E6100000	4.8858943	52.300198
285	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:32-04	0101000020E6100000	4.8858943	52.300198
286	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:42-04	0101000020E6100000	4.896596	52.2976
287	8775680	17	IN TRANSIT	34078	2014-05-28 13:04:52-04	0101000020E6100000	4.896596	52.2976
288	8775680	18	STOPPED AT	0	2014-05-28 13:05:02-04	0101000020E6100000	4.897852	52.297974
289	8775680	18	STOPPED AT	0	2014-05-28 13:05:12-04	0101000020E6100000	4.897852	52.297974
290	8775680	18	STOPPED AT	0	2014-05-28 13:05:22-04	0101000020E6100000	4.897852	52.297974
291	8775680	18	IN TRANSIT	45269	2014-05-28 13:05:32-04	0101000020E6100000	4.897852	52.297974

extract of recorded vehiclePositions for a selected trip

	trip_id integer	stop_id integer	arrival_time text	departure_time text	stop_sequence smallint
1	8775680	45550	18:22:00	18:22:00	0
2	8775680	41898	18:26:00	18:26:00	4
3	8775680	38058	18:27:00	18:27:00	5
4	8775680	38122	18:29:00	18:29:00	6
5	8775680	36760	18:32:00	18:32:00	7
6	8775680	49694	18:34:00	18:34:00	8
7	8775680	49681	18:35:00	18:35:00	9
8	8775680	32887	18:50:00	18:51:00	13
9	8775680	36200	18:52:00	18:52:00	14
10	8775680	46529	18:53:00	18:53:00	15
11	8775680	42461	18:54:00	18:54:00	16
12	8775680	34078	18:54:00	18:54:00	17
13	8775680	45269	18:58:00	18:58:00	18
14	8775680	34752	18:58:00	18:58:00	19
15	8775680	47382	19:00:00	19:00:00	20
16	8775680	132250	19:02:00	19:02:00	22
17	8775680	132204	19:02:00	19:02:00	24
18	8775680	168481	19:03:00	19:03:00	25
19	8775680	168465	19:06:00	19:06:00	26

original timetable

	trip_id integer	stop_id integer	arrival_time text	departure_time text	stop_sequence smallint
1	8775680	45550	18:02:32	18:06:12	0
2	8775680	41898	18:26:02	18:26:12	4
3	8775680	38058	18:27:12	18:27:12	5
4	8775680	38122	18:28:22	18:28:52	6
5	8775680	36760	18:30:42	18:30:42	7
6	8775680	49694	18:36:12	18:36:12	8
7	8775680	49681	18:37:52	18:38:32	9
8	8775680	32887	18:58:32	18:58:42	13
9	8775680	36200	18:59:32	18:59:32	14
10	8775680	46529	19:00:32	19:00:32	15
11	8775680	42461	19:01:12	19:01:32	16
12	8775680	34078	19:02:22	19:02:42	17
13	8775680	45269	19:05:02	19:05:22	18
14	8775680	34752	19:06:02	19:06:02	19
15	8775680	47382	19:06:52	19:06:52	20
16	8775680	132250	19:08:55	19:08:55	22
17	8775680	132204	19:09:45	19:09:55	24
18	8775680	168481	19:11:05	19:11:05	25
19	8775680	168465	19:14:05	19:14:05	26

modified timetable

Fig. 33: From given timetable to modified timetable using recorded vehiclePositions

6.4 Delay analysis

For this part of analysis, the delay information of every single stop in all available trips needed to be calculated. To do this, the difference of all observed times compared to static times was used to create a frequency histogram. During the task, huge outliers were detected because of data quality issues. The reason is, for instance, that the stop sequence hasn't been followed in the correct sequence, or some stops were not recorded until a few hours after the planned arrival time. Therefore, an appropriate time range of delays between -1800 and +1800 seconds was defined, whereas all other delays were ignored for further analysis. In Figure 34 it is visible that there are enormous delays below that level, which would falsify the output.

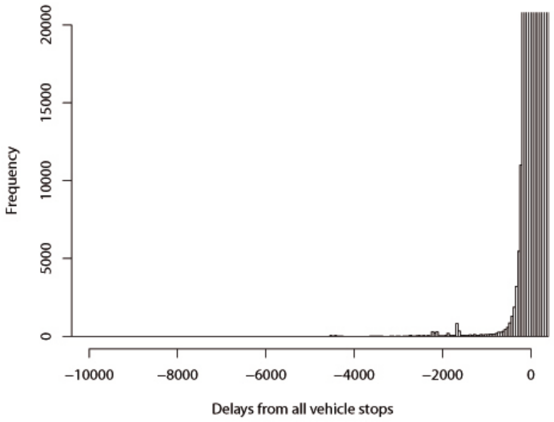

Fig. 34: Distribution of outliers in frequency histogram

The distribution of delay information in the pre-defined time range is given in Figure 35. The different classes from 0 to 40 describe the - 100 seconds - steps from -1800 to 1800 seconds and can be found in the appendix. It is noticeable that the highest amount of stop times has a delay between 1 and 100 seconds (class 20), followed by class 21 with about 300.000 stop times in the range between 101 and 200 seconds.

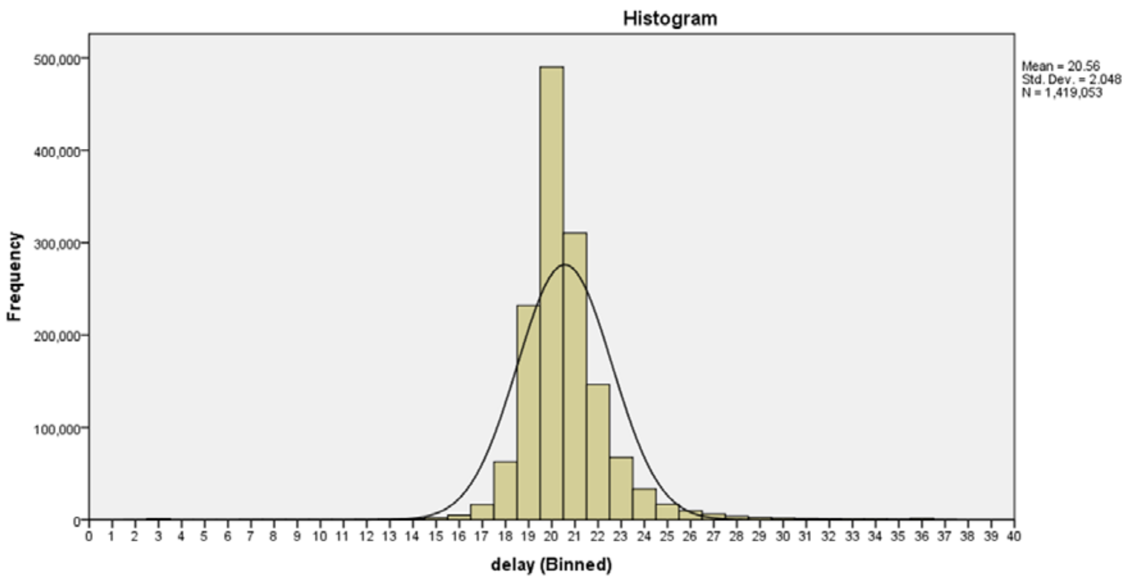

Fig. 35: Histogram of all delays (< 1800s and >-1800s) calculated in time range between 07:00 and 20:00

Additionally, all delays were aggregated to their dedicated stops to calculate average delays for each stop. In Figure 36, altogether 41.218 stops (out of 72.256 from the whole network) contain an average delay information, whereas most of the stops contain an average delay of 0 to 99 seconds.

labels	# of average delay affected stops
<-1500 oder (Leer)	1
-1500--1401	1
-1400--1301	1
-1300--1201	1
-1200--1101	1
-1000--901	1
-800--701	2
-700--601	5
-600--501	13
-500--401	16
-400--301	39
-300--201	90
-200--101	377
-100--1	2932
0-99	16004
100-199	15114
200-299	4896
300-399	1305
400-499	228
500-599	75
600-699	33
700-799	15
800-899	1
900-999	15
1000-1099	19
1100-1199	15
1200-1299	8
1300-1399	10
total count	**41218**

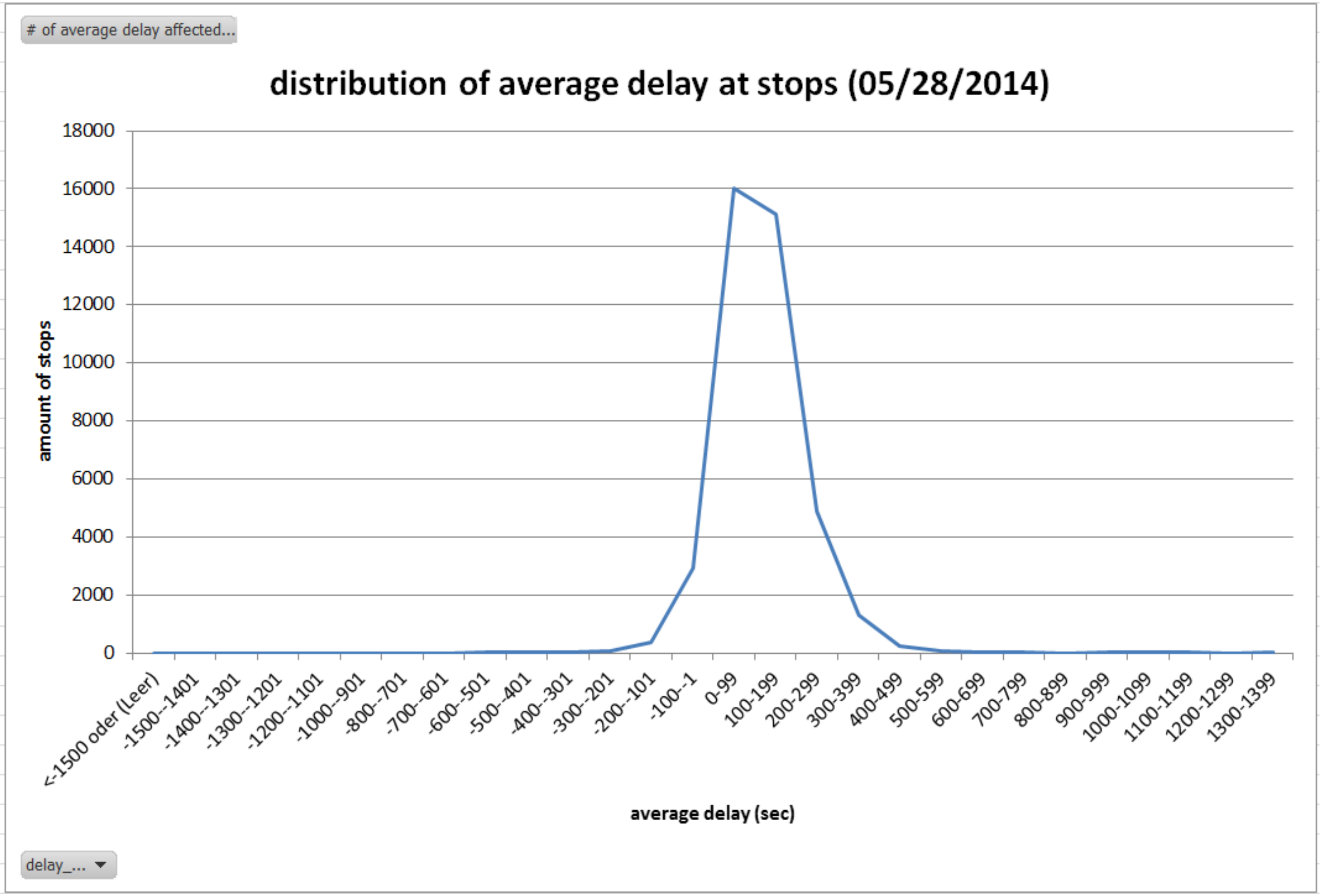

Fig. 36: Histogram of average delay at stops

Using average delays, it is also possible to split the information into single hours of a day. In Figure 37, the average delay between 11:00 and 12:00 at stops is highlighted. Altogether, 36655 were affected by delays during this time period, which is about half the amount of all stops in the network.

labels	# of average delay affected stops
<-1500 oder (Leer)	2
-1500--1401	1
-1400--1301	1
-1200--1101	3
-1100--1001	4
-1000--901	4
-900--801	8
-800--701	2
-700--601	8
-600--501	10
-500--401	13
-400--301	34
-300--201	158
-200--101	991
-100--1	5050
0-99	14321
100-199	10570
200-299	3846
300-399	1125
400-499	277
500-599	123
600-699	41
700-799	30
800-899	14
900-999	5
1000-1099	2
1100-1199	3
1200-1299	3
1300-1399	1
1400-1499	2
1500-1599	1
1600-1699	1
1700-1799	1
total count	**36655**

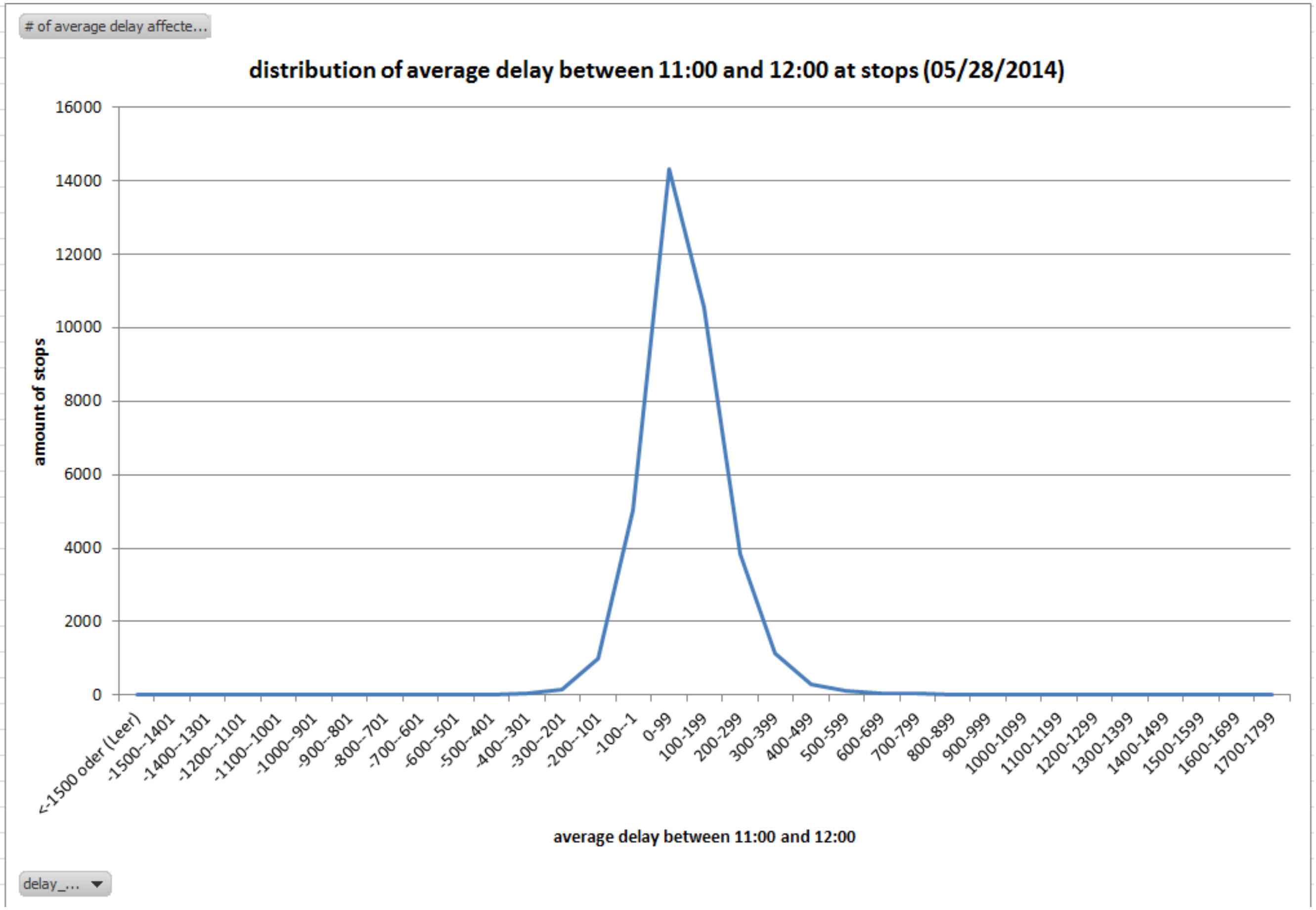

Fig. 37: Histogram of average delays between 11:00 and 12:00 at stops

In Figure 38, a histogram of average delays between 18:00 and 19:00 is visualized. It is noticeable that the total amount of affected stops is slightly lower than in Figure 36.

labels	# of average delay affected stops
<-1000 oder (Leer)	1
-900--801	2
-800--701	2
-700--601	4
-600--501	7
-500--401	13
-400--301	44
-300--201	235
-200--101	1163
-100--1	5222
0-99	11261
100-199	6945
200-299	3015
300-399	1652
400-499	906
500-599	538
600-699	377
700-799	254
800-899	166
900-999	73
1000-1099	62
1100-1199	53
1200-1299	32
1300-1399	19
1400-1499	17
1500-1599	5
1600-1699	5
1700-1799	11
total count	**32084**

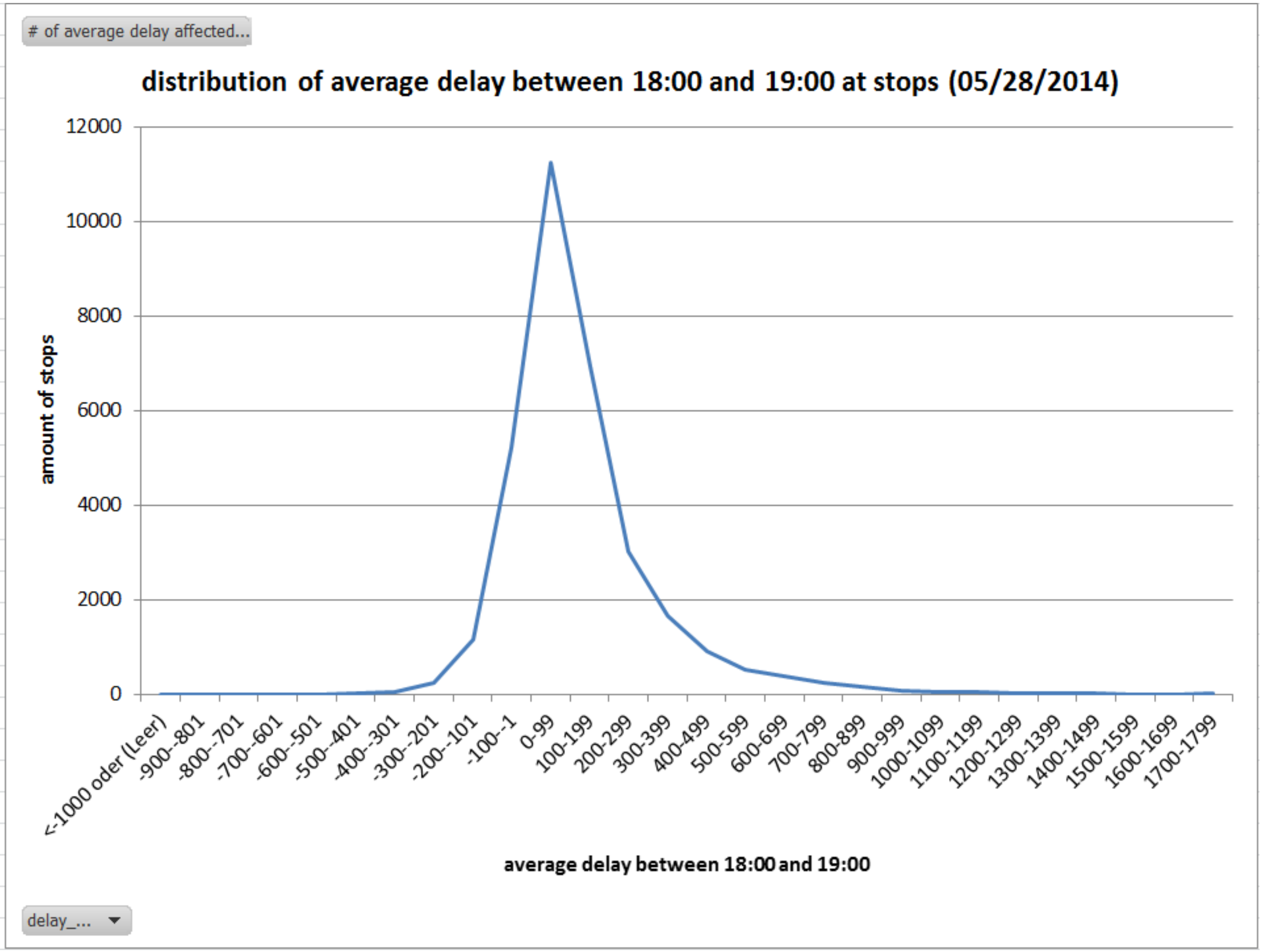

Fig. 38: Histogram of average delays between 18:00 and 19:00 at stops

Besides, it is also interesting to take a look at the average delay of a single stop. In Figure 39, the hourly average delay from 7:00 until 20:00 is displayed. The peak of highest delays at this stop occurred between 9:00 and 10:00 with more than 180 average seconds. The lowest average delay was between 18:00 and 19:00 with a mean of about 25 seconds.

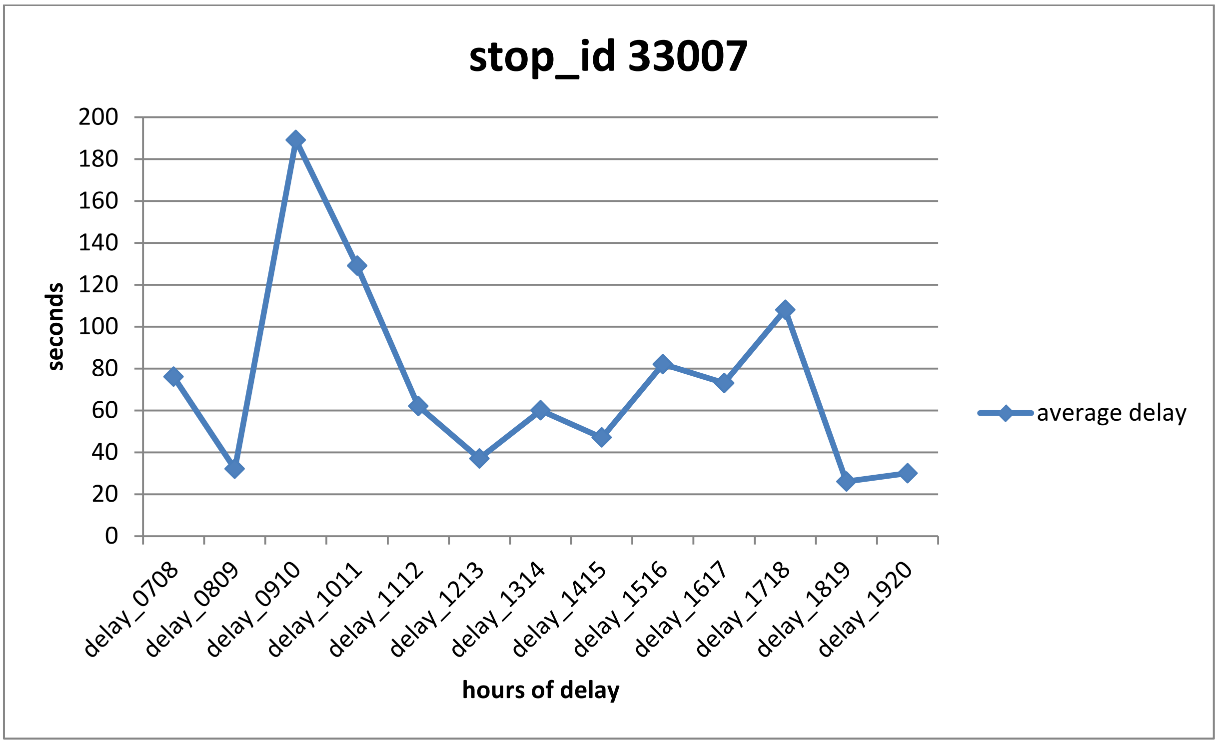

Fig. 39: Hourly average delays at stop 33007

Figure 40 shows a map of all stops which were affected by delay information in the period of time the data was collected. As interpolation method, kriging was chosen to highlight the areas where delays occurred. Using the average delay of each stop as magnitude value, the most affected areas are the central and the northern parts of Netherlands.

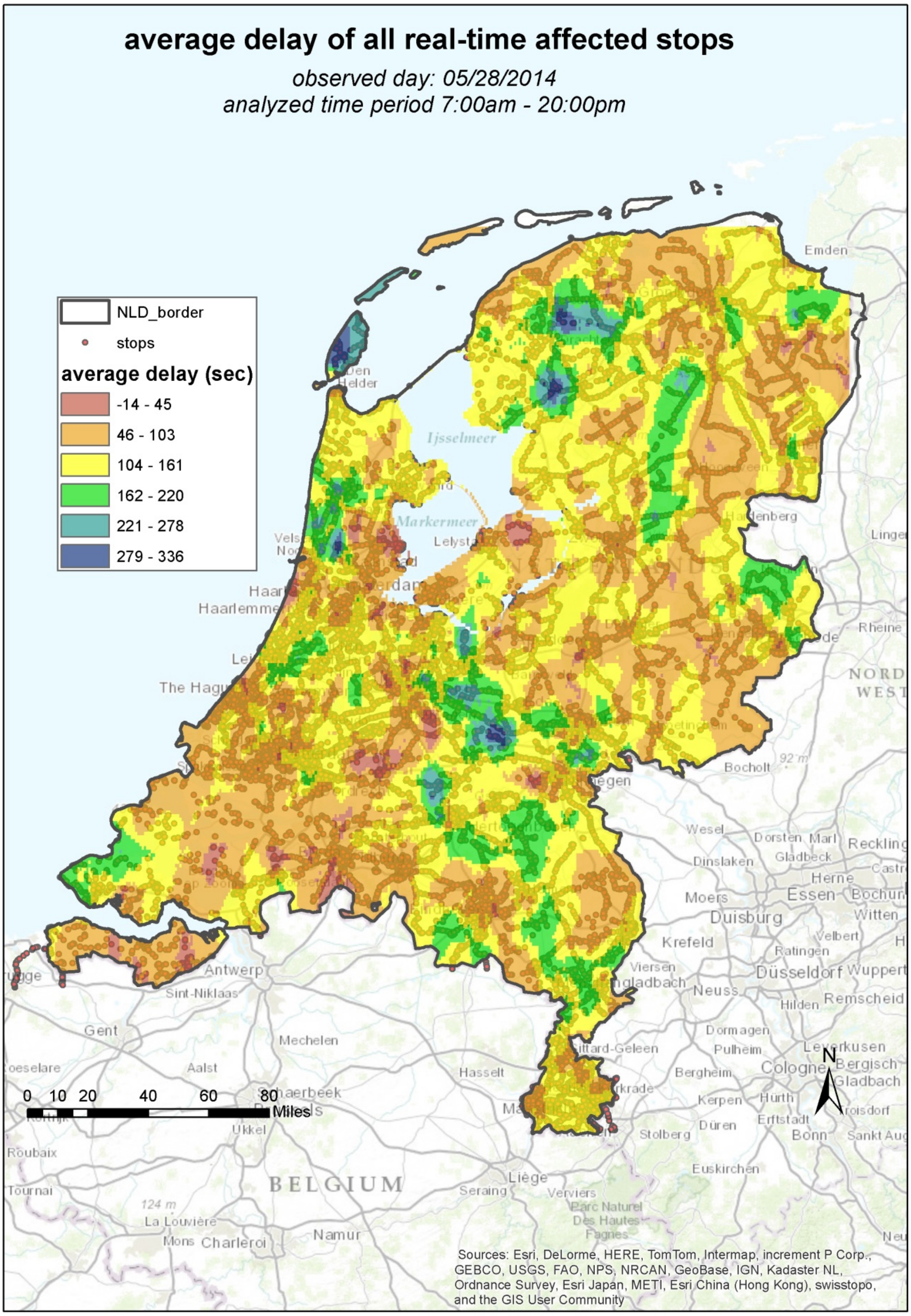

Fig. 40: Stops that were affected by an average delay

6.5 Impact of tripUpdates

As a second data source of realtime information, tripUpdates were recorded every 30 seconds and saved into the database. Additionally, all downloaded files were stored locally using timestamp information, to be able to use the feeds during the route calculation in scenario 2. TripUpdates contain estimated delay information for future stops as well as for past stops. In the following two diagrams (Figure 41 & 42), the delay for all stops on a particular trip is visualized. In both cases, the information of the green line was extracted from the tripUpdate file that contained that trip for the first time. The purple line consists of information from the tripUpdate feed that was collected during the trip and the blue line refers to information recorded after the trip ended. Therefore the information from before the actual timestamp is not necessary for trip planning purposes any more. Besides, the red line refers to real (observed) delay information, extracted from vehiclePositions data.

These samples show that there is an enormous lack of data quality for the passengers. First of all, the tripUpdate information is not available until the trip has started. After it started, delay information for the past was similar to observed delays, but future delays were hardly available or just estimated to get zero at the end of the trip. After the trip has finished, the delay information was very similar compared to the observed delay, but didn't have any effects on trip planning any more.

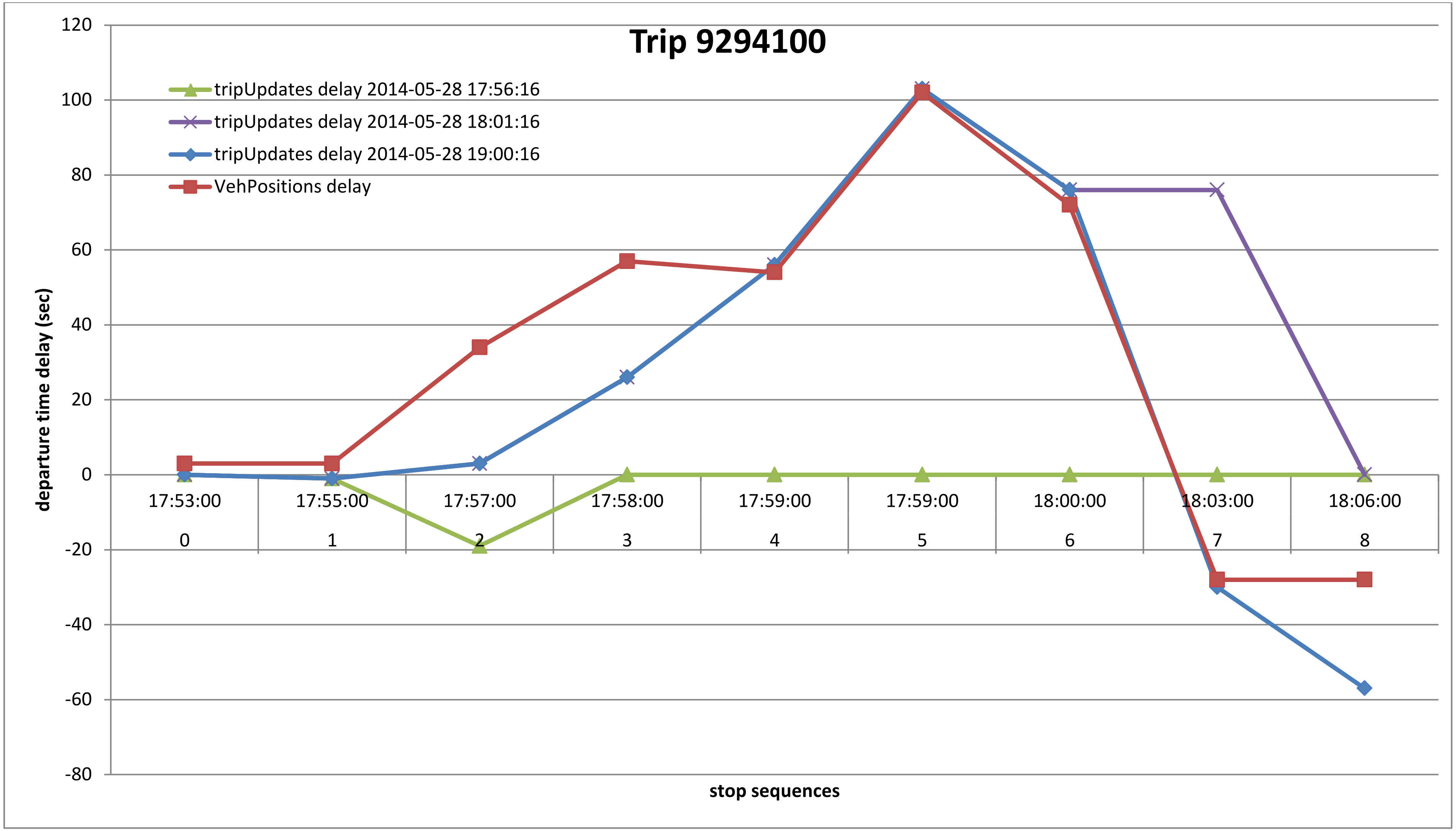

Fig. 41: Comparing delay predictions from tripUpdates and observed delays from vehiclePositions of trip 9294100

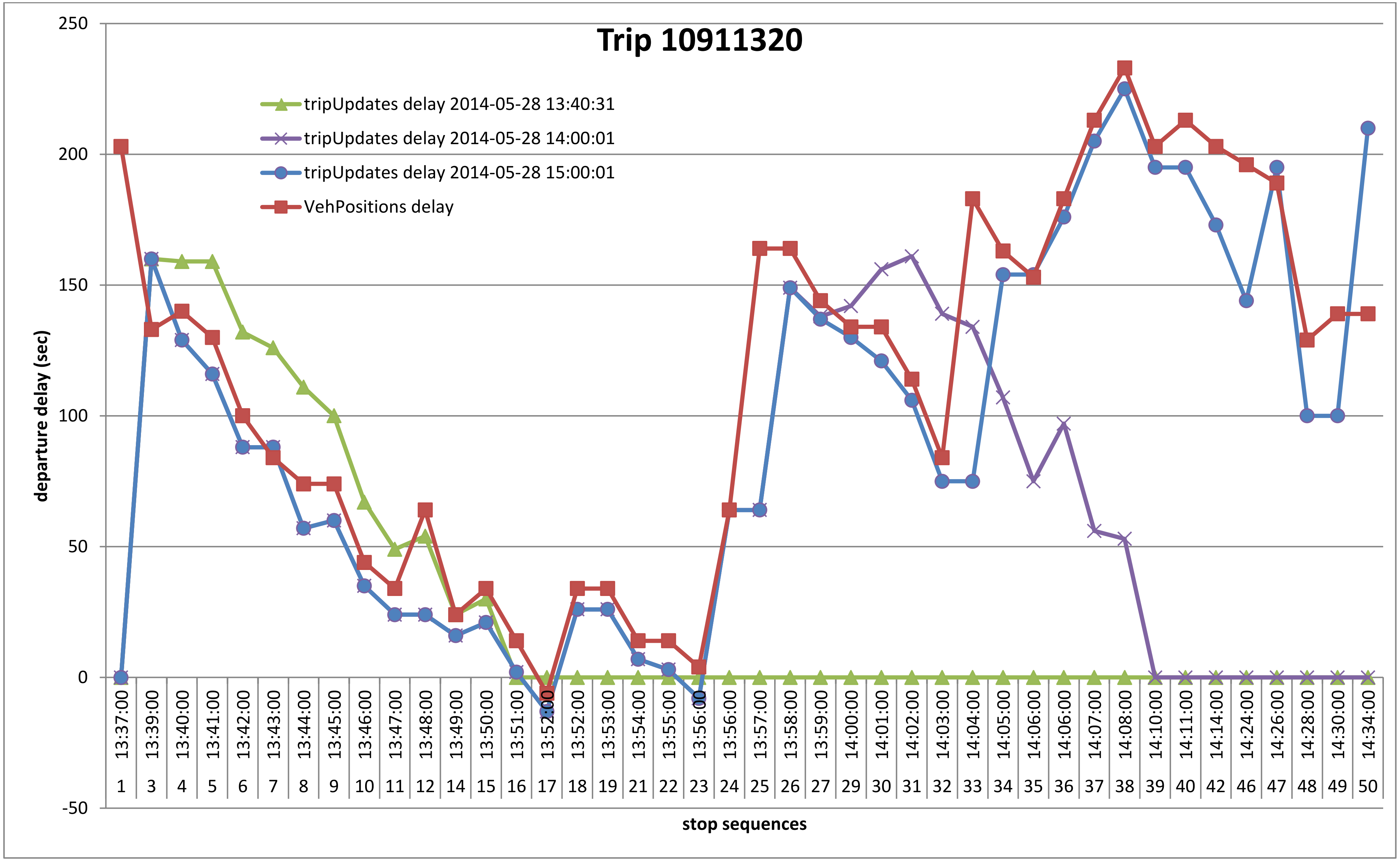

Fig. 42: Comparing delay predictions from tripUpdates and observed delays from vehiclePositions of trip 10911320

6.6 Testing different scenarios

6.6.1 Implementation

For this part of the research, an ArcGIS model (Figure 43) has been created that generates 45 random points out of a pre-defined area in Netherlands (extract). The rules defined in the model only allow random points to be in a buffer which is up to 300 meters around a bus route (shapelines) and 300 meters around a stop (gtfs_stops). To get 45 random origins and 45 random destinations, the model was executed twice. The coordinates were used to create 45 random routes for all scenarios at a time. All the routes were planned for May 28th, 9:00:00 am, with a maximum walking distance of 2500m and the OTP route mode "Busish&Walk".

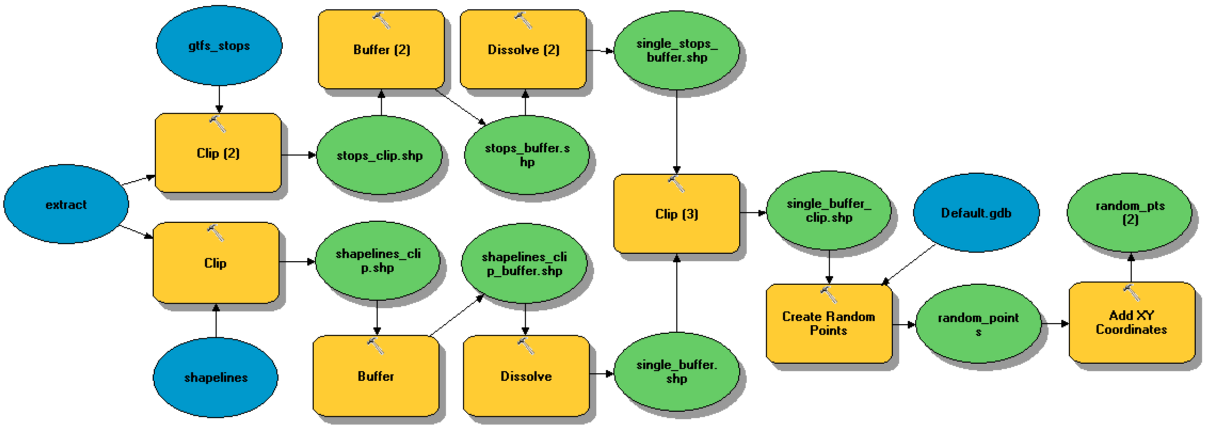

Fig. 43: ArcGIS model to create random points

In Figure 44, the extracted area and the created random points in this area are visible. The reason why this shape in Central Netherlands was defined is that it is supposed to contain most of the real-time affected districts.

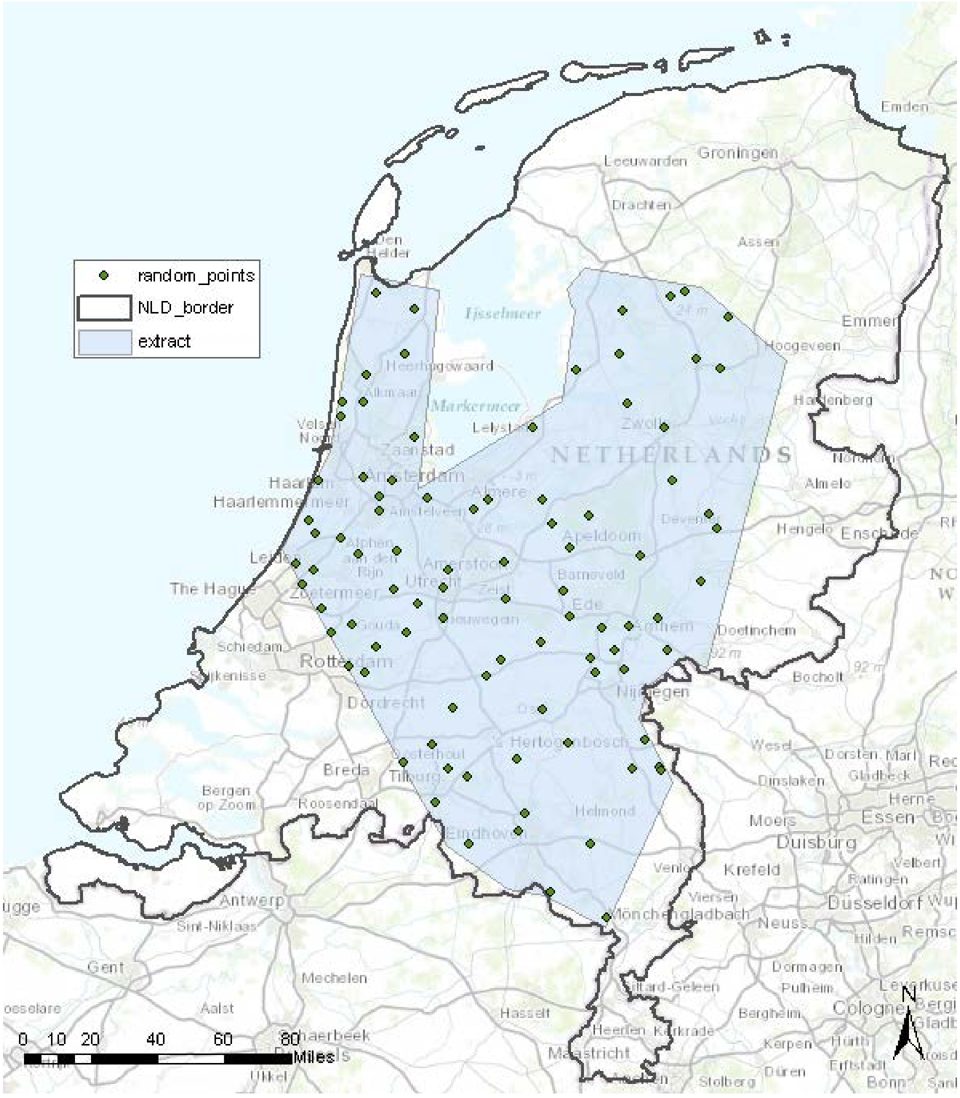

Fig. 44: Defined area to create random points

6.6.2 Considered scenarios

<u>1. scenario (route printed on paper, no real-time delay predictions):</u>

In the first scenario, as described in section 4.5.4, the passenger initially calculates the route from a given origin, destination and time. A route is displayed on the screen and the user prints the paper and follows the route according to the calculated schedule. As soon as the passenger misses a bus at a stop, he waits for the following bus that is going to drive exactly the same route to the next suggested stop. The algorithm works in the way that the static arrival and departure times from every suggested bus on the calculated route will be compared to the observed times that are stored in the reference table (modified stop_times). In case a bus departed before the passenger arrived there, the following bus which is going to drive exactly the same route than the missed one, will be inquired. Walking times from the initial response will be added to actual times after the bus arrived. This method is repeated until the passenger reaches his destination. Compared to scenario 3, this one is not recursive, so no new REST request will be sent.

Figure 45 shows an example of a calculated route in scenario 1. The first 14 rows (num_calculation = 1) represent the calculated trip from the REST response using static stop_times from the GTFS feed. The column “count” claims that it was the 26th route out of 45. “Calc_type” 1 implies the scenario that was chosen. In the second phase of the calculation (num_calculation = 2), the algorithm starts from the first trip and compares the start_time (=departure time) as well as end_time (=arrival time) with observed times from the modified stop_times table. Up to row 25, no bus would have been missed. Since trip 11373739 arrived at 11:59:47 instead of 11:59:00, the passenger would have missed this bus, so the following bus (11379754) that departed at 12:00:00 was chosen. It arrived at 12:09:00, whereas the amount of time for the following walking distance was taken from the initial calculation and added to that one. Altogether, the trip itself would have been about 2 minutes faster than the initial calculation.

	trip_id integer	start_time time without time zone	end_time time without time zone	route_mode text	num_calculation smallint	count smallint	calc_type smallint
1		09:02:33	09:05:09	WALK	1	26	1
2	12507376	09:05:10	09:14:00	BUS	1	26	1
3	12500405	09:40:00	09:54:27	BUS	1	26	1
4		09:54:27	09:54:49	WALK	1	26	1
5	11515254	10:00:00	10:07:00	BUS	1	26	1
6	10393701	10:10:00	10:45:00	BUS	1	26	1
7		10:45:00	10:45:02	WALK	1	26	1
8	8800548	11:13:00	11:32:00	BUS	1	26	1
9		11:32:00	11:32:18	WALK	1	26	1
10	11373739	11:42:00	11:59:00	BUS	1	26	1
11	11372865	11:59:00	12:07:00	BUS	1	26	1
12		12:07:00	12:07:11	WALK	1	26	1
13	10389450	12:25:00	13:02:00	BUS	1	26	1
14		13:02:01	13:13:22	WALK	1	26	1
15		09:06:39	09:09:15	WALK	2	26	1
16	12507376	09:09:16	09:16:26	BUS	2	26	1
17	12500405	09:37:46	09:53:57	BUS	2	26	1
18		09:53:58	09:54:20	WALK	2	26	1
19	11515254	10:02:37	10:08:07	BUS	2	26	1
20	10393701	10:09:37	10:44:57	BUS	2	26	1
21		10:44:58	10:45:00	WALK	2	26	1
22	8800548	11:13:17	11:32:57	BUS	2	26	1
23		11:32:58	11:33:16	WALK	2	26	1
24	11373739	11:44:27	11:59:47	BUS	2	26	1
25	11379754	12:00:00	12:09:00	BUS	2	26	1
26		12:09:01	12:09:12	WALK	2	26	1
27	10389450	12:26:27	13:00:08	BUS	2	26	1
28		13:00:09	13:11:30	WALK	2	26	1

Fig. 45: Route example from scenario 1

2. scenario (route printed on paper, including real-time delay predictions):

This scenario is based on the first one and additionally considers real-time delay predictions that are given in the tripUpdate feed, which is added to the OTP routing engine. Here, future delays on selected trips that are served as real-time information during the route calculation, may affect the trip time period. For this work, the tripUpdate feed from May 28th, 09:00:00 am, was archived to be used for this scenario. As described in section 1, the tripUpdate feed from that timestamp contains 10451 different trips, which is about 1,5 % of all trips in the feed. When taking a look at the diagrams from chapter 6.5, the chance of a delay in a particular section of a calculated trip decreases significantly. In the simulation using the 45 routes, altogether 23 trips out of 596 trips were affected by real-time information. In Figure 46, it is also visible that only 4 of all real_time affected trips actually had a delay (=0,7% of all calculated trips).

	trip_id integer	start_time time without time zone	end_time time without time zone	route_mode text	arr_delay integer	dep_delay integer	real_time boolean	num_calculation smallint	count smallint	calc_type smallint
1	8798774	09:07:48	09:23:00	BUS	0	168	t	1	3	2
2	8747157	09:12:00	09:29:00	BUS	0	0	t	1	5	2
3	9039756	09:24:00	09:51:00	BUS	0	0	t	1	6	2
4	8820779	09:05:00	09:28:00	BUS	0	0	t	1	7	2
5	11843236	09:22:00	09:45:00	BUS	0	0	t	1	8	2
6	8751720	09:18:00	09:22:00	BUS	0	0	t	1	9	2
7	11524835	09:03:00	09:25:00	BUS	0	0	t	1	13	2
8	11213895	09:03:12	09:08:00	BUS	0	132	t	1	16	2
9	11207932	09:11:00	09:32:00	BUS	0	0	t	1	16	2
10	11497662	09:11:00	09:44:00	BUS	0	0	t	1	21	2
11	12081282	09:05:00	09:31:00	BUS	0	0	t	1	18	2
12	12507376	09:08:21	09:17:36	BUS	30	191	t	1	26	2
13	12500405	09:41:36	09:54:27	BUS	0	0	t	1	26	2
14	11520794	09:09:00	09:51:00	BUS	0	0	t	1	23	2
15	10091828	09:26:00	09:31:00	BUS	0	0	t	1	24	2
16	9281834	09:07:00	09:16:00	BUS	0	0	t	1	25	2
17	11952242	09:20:00	10:03:00	BUS	0	0	t	1	30	2
18	8776575	09:11:00	09:19:00	BUS	0	0	t	1	32	2
19	10400329	09:26:00	09:56:00	BUS	0	0	t	1	32	2
20	12163996	09:08:00	10:05:00	BUS	0	0	t	1	38	2
21	8815139	09:09:12	09:49:00	BUS	0	72	t	1	39	2
22	6065151	09:10:00	09:16:00	BUS	0	0	t	1	41	2
23	11520794	09:32:00	09:51:00	BUS	0	0	t	1	40	2

Fig. 46: Real-time affected trips during the calculation of 45 route for scenario 2

Figure 47 visualizes an extract of the calculated route 26 and gives information about how the delay information affected the length of the route (compared to scenario 1 in Figure 45).

	trip_id integer	start_time time without time zone	end_time time without time zone	route_mode text	arr_delay integer	dep_delay integer	real_time boolean	num_calculation smallint	count smallint	calc_type smallint
1		09:05:44	09:08:20	WALK				1	26	2
2	507376	09:08:21	09:17:36	BUS	30	191	t	1	26	2
3	500405	09:41:36	09:54:27	BUS	0	0	t	1	26	2
4		09:54:27	09:54:49	WALK				1	26	2
5	515254	10:00:00	10:07:00	BUS	0	0	f	1	26	2
6	393701	10:10:00	10:45:00	BUS	0	0	f	1	26	2
7		10:45:00	10:45:02	WALK				1	26	2
8	800548	11:13:00	11:32:00	BUS	0	0	f	1	26	2
9		11:32:00	11:32:18	WALK				1	26	2
10	373739	11:42:00	11:59:00	BUS	0	0	f	1	26	2
11	372865	11:59:00	12:07:00	BUS	0	0	f	1	26	2
12		12:07:00	12:07:11	WALK				1	26	2
13	389450	12:25:00	13:02:00	BUS	0	0	f	1	26	2
14		13:02:01	13:13:22	WALK				1	26	2
15		09:06:39	09:09:15	WALK				2	26	2
16	507376	09:09:16	09:20:36	BUS				2	26	2
17	500405	09:44:57	09:53:57	BUS				2	26	2
18		09:53:58	09:54:20	WALK				2	26	2
19	515254	10:02:37	10:08:07	BUS				2	26	2
20	393701	10:09:37	10:44:57	BUS				2	26	2
21		10:44:58	10:45:00	WALK				2	26	2
22	800548	11:13:17	11:32:57	BUS				2	26	2
23		11:32:58	11:33:16	WALK				2	26	2
24	373739	11:44:27	11:59:47	BUS				2	26	2
25	379754	12:00:00	12:09:00	BUS				2	26	2
26		12:09:01	12:09:12	WALK				2	26	2
27	389450	12:26:27	13:00:08	BUS				2	26	2
28		13:00:09	13:11:30	WALK				2	26	2

Fig. 47: Route example from scenario 2, including delay information

3. scenario (Smartphone, no real-time delay predictions):

This scenario differs from the first ones, as the passenger uses a smartphone for the route calculation. Because of this, he is not dependent on following the route on the printed map. If he misses a bus, it is possible to request a new route to the destination, based on the current position. In the simulation of that scenario, this algorithm is built recursively. It means that as soon as the case of a missed bus happens, a new route will be calculated from the current stop the passenger missed the bus. The recalculation is visible in Figure 47 in the column num_calcul. The number one in this column is the initial route that was calculated by OTP. The higher values give information in case the bus has been missed. After the initial route calculation, the arrival and departure times of each bus will be compared with the modified stop_times table that includes the observed timetables. For instance, the first bus with tripID 12492038 had an observed departure time of 08:30:28 and an arrival time of 09:23:46. Adding the time of the walking trip plus the delay of the bus, the passenger would have missed the following bus 11515352, who departed at 09:23:56. So a new route had to be calculated, based on the current position at that stop. The problem in this scenario is that the OTP only uses static time schedules for recalculation and is not aware of the observed stop_times. This is what Figure 48 represents in the highlighted trips. After the algorithm was aware of the missed bus 12731416, a new route has been calculated from that stop point. Since the OTP wasn't aware about observed times from trip 12731416, the same bus was taken like in the initial calculation. Because of this fact that the OTP is not able to receive information about observed times from the modified stop_times table, inconsistences occur in the output where the following bus starts earlier than the bus before. Thus, the last two lines remain inconsistent.

	trip_id	calc_type	start_time	end_time	timestmp	route_mode	num_calcul
0	NULL	3	08:26:05	08:28:59	05:38:57	WALK	1
1	12492038	3	08:29:00	09:19:25	05:38:57	BUS	1
2	NULL	3	09:19:25	09:19:45	05:38:57	WALK	1
3	11515352	3	09:20:00	09:49:00	05:38:57	BUS	1
4	NULL	3	09:49:00	09:49:02	05:38:57	WALK	1
5	11515704	3	10:20:00	10:59:00	05:38:57	BUS	1
6	NULL	3	10:59:00	10:59:18	05:38:57	WALK	1
7	11852222	3	11:06:00	12:44:00	05:38:57	BUS	1
8	NULL	3	12:44:00	12:46:20	05:38:57	WALK	1
9	12715960	3	13:00:00	13:54:00	05:38:57	BUS	1
10	12731416	3	13:54:00	14:45:00	05:38:57	BUS	1
11	NULL	3	14:45:01	14:49:31	05:38:57	WALK	1
12	NULL	3	08:26:05	08:28:59	05:38:57	WALK	2
13	12492038	3	08:30:28	09:23:46	05:38:57	BUS	2
14	NULL	3	09:23:47	09:24:07	05:38:57	WALK	2
15	NULL	3	09:44:37	09:49:59	05:38:57	WALK	3
16	11515346	3	09:51:07	10:14:07	05:38:57	BUS	3
17	NULL	3	10:14:08	10:14:10	05:38:57	WALK	3
18	11515704	3	10:20:17	10:58:17	05:38:57	BUS	3
19	NULL	3	10:58:18	10:58:36	05:38:57	WALK	3
20	11852222	3	11:07:47	12:41:47	05:38:57	BUS	3
21	NULL	3	12:41:48	12:44:08	05:38:57	WALK	3
22	12715960	3	12:59:58	13:53:34	05:38:57	BUS	3
23	12731416	3	13:53:24	14:43:54	05:38:57	BUS	4
24	NULL	3	14:45:01	14:49:31	05:38:57	WALK	4

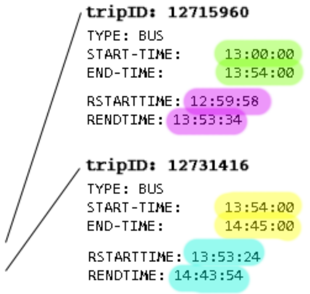

Fig. 48: Extract from a route calculated for scenario 3

4. scenario (Smartphone, including real-time predictions):

This scenario is based on the third one, where the algorithm simulates a passenger who calculates a new route with his smartphone as soon as a bus was missed. Additionally, the tripUpdates is included every time a route is being calculated. The "simulation" of real-time route calculation is, as described in scenario 3, not possible with this version of the OpenTripPlanner, since it is not aware of the observed stop times from the table that will be used for scenario 1 and 2.

5. scenario (observed arrival and departure times contained in the routing graph):

The fifth scenario makes use of a graph that contains the observed arrival and departure times that were gained through the collected vehiclePositions in the 13-hour-period from May 28th, 2014. The modified stop_times (described in 5.3.3) were exported into the GTFS feed, which was used to create the routing graph in the OpenTripPlanner. All observed times are considered to be the "optimal" schedule times the passenger could have taken. Furthermore, this means that all trips that are calculated in this scenario are supposed to result in the "optimal" route the passenger could have taken. In this scenario, the initial route calculation (num_calculation = 1) already includes the "real" arrival and departure times (Figure 49).

	trip_id integer	start_time time without time zone	end_time time without time zone	route_mode text	num_calculation smallint	count smallint	calc_type smallint
1		09:06:39	09:09:15	WALK	1	26	5
2	12507376	09:09:16	09:20:36	BUS	1	26	5
3	12500405	09:44:57	09:53:57	BUS	1	26	5
4		09:53:57	09:54:19	WALK	1	26	5
5	11515254	10:02:37	10:08:07	BUS	1	26	5
6	10393701	10:09:37	10:44:57	BUS	1	26	5
7		10:44:57	10:44:59	WALK	1	26	5
8	8800548	11:13:17	11:32:57	BUS	1	26	5
9		11:32:57	11:33:15	WALK	1	26	5
10	11373739	11:44:27	11:54:57	BUS	1	26	5
11	11372865	11:54:57	12:08:17	BUS	1	26	5
12		12:08:17	12:08:28	WALK	1	26	5
13	10389450	12:26:27	13:00:08	BUS	1	26	5
14		13:00:09	13:11:30	WALK	1	26	5

Fig. 49: Extract from a route calculated for scenario 5

In Figure 50, the map shows the calculated route 26 from the scenarios 1, 2 and 5, whereas there is no difference visible.

Fig. 50: Visualization of the calculated route 26 in scenarios 1, 2 and 5

6.6.3 Comparison of results

In Figure 51, the result of the three different scenarios 1, 2 and 5 is given. The table to the right shows time savings in seconds between different scenarios. Although scenario 5 is supposed to have equal or fastest trip times compared to scenario 1, only 69 % of all routes are faster. The reason could be that the OTP is not able to handle the GTFS feed with the modified stop_times.

	scenario 1						scenario 2						scenario 5			seconds		
	1st run			2nd run			1st run			2nd run			1st run			time savings in sc2	time savings in sc2	time savings in sc5
route	start	end	**duration**	start	end	**duration**	start	end	**duration**	start	end	**duration**	start	end	**duration**	diff sc1-sc2 (1st run)	diff sc1-sc2 (2nd run)	diff sc5 and sc1 (2nd run)
0	09:01:40	14:28:39	**05:26:59**	09:01:40	14:28:39	**05:26:59**	09:01:40	14:28:39	**05:26:59**	09:01:40	14:28:39	**05:26:59**	09:01:40	14:28:39	**05:26:59**	0	0	0
1	09:20:36	13:07:01	**03:46:25**	09:20:36	13:11:59	**03:51:23**	09:20:36	13:07:01	**03:46:25**	09:20:36	13:11:59	**03:51:23**	09:27:32	13:11:59	**03:44:27**	0	0	-416
2	09:24:44	13:54:37	**04:29:53**	09:24:44	13:54:37	**04:29:53**	09:24:44	13:54:37	**04:29:53**	09:24:44	13:54:37	**04:29:53**	09:25:00	13:54:37	**04:29:37**	0	0	-16
3	09:01:28	11:54:09	**02:52:41**	09:01:28	12:24:09	**03:22:41**	09:04:16	11:54:09	**02:49:53**	09:04:16	12:24:09	**03:19:53**	09:46:15	13:24:09	**03:37:54**	-168	-168	913
4	09:12:51	14:13:08	**05:00:17**	09:12:51	14:13:08	**05:00:17**	09:12:51	14:13:08	**05:00:17**	09:12:51	14:13:08	**05:00:17**	09:42:18	18:13:08	**08:30:50**	0	0	12633
5	09:07:42	12:08:36	**03:00:54**	09:07:42	12:10:53	**03:03:11**	09:07:42	12:08:36	**03:00:54**	09:07:42	12:10:53	**03:03:11**	09:07:48	12:10:53	**03:03:05**	0	0	-6
6	09:14:46	13:17:47	**04:03:01**	09:14:46	13:17:47	**04:03:01**	09:14:49	13:17:47	**04:02:58**	09:14:49	13:17:47	**04:02:58**	09:05:13	15:12:57	**06:07:44**	-3	-3	7483
7	09:04:23	14:21:07	**05:16:44**	08:54:26	14:45:07	**05:50:41**	09:04:23	14:21:07	**05:16:44**	08:54:26	14:45:07	**05:50:41**	09:03:37	14:21:07	**05:17:30**	0	0	-1991
8	09:20:12	13:32:47	**04:12:35**	09:20:12	13:32:47	**04:12:35**	09:20:12	13:32:47	**04:12:35**	09:20:12	13:32:47	**04:12:35**	09:21:28	13:45:03	**04:23:35**	0	0	660
9	09:09:23	15:01:29	**05:52:06**	09:09:23	16:01:19	**06:51:56**	09:09:23	15:01:29	**05:52:06**	09:09:23	16:01:19	**06:51:56**	09:12:09	15:11:53	**05:59:44**	0	0	-3132
10	09:08:49	13:30:12	**04:21:23**	09:08:49	13:45:12	**04:36:23**	09:08:49	13:30:12	**04:21:23**	09:08:49	13:45:12	**04:36:23**	09:08:49	13:30:30	**04:21:41**	0	0	-882
11	09:09:10	10:09:35	**01:00:25**	09:09:10	11:09:35	**02:00:25**	09:09:10	10:09:35	**01:00:25**	09:09:10	11:09:35	**02:00:25**	09:09:06	10:10:42	**01:01:36**	0	0	-3529
12	10:07:18	15:40:49	**05:33:31**	10:00:07	15:41:17	**05:41:10**	10:07:18	15:40:49	**05:33:31**	10:00:07	15:41:17	**05:41:10**	09:07:18	15:41:17	**06:33:59**	0	0	3169
13	09:00:38	14:08:07	**05:07:29**	09:00:38	15:12:13	**06:11:35**	09:00:38	14:08:07	**05:07:29**	09:00:38	15:12:13	**06:11:35**	09:50:13	14:39:16	**04:49:03**	0	0	-4952
14	10:36:30	14:24:35	**03:48:05**	10:36:30	14:24:35	**03:48:05**	10:36:30	14:24:35	**03:48:05**	10:36:30	14:24:35	**03:48:05**	09:00:05	12:57:31	**03:57:26**	0	0	561
15	09:31:56	15:15:02	**05:43:06**	09:31:56	15:16:38	**05:44:42**	09:31:56	15:15:02	**05:43:06**	09:31:56	15:16:38	**05:44:42**	09:31:56	15:16:38	**05:44:42**	0	0	0
16	09:58:28	12:08:31	**02:10:03**	09:58:28	13:08:31	**03:10:03**	09:00:40	11:08:31	**02:07:51**	09:00:40	11:06:37	**02:05:57**	09:00:04	11:38:27	**02:38:23**	-132	-3846	-1900
17	09:27:31	12:19:44	**02:52:13**	09:27:31	12:21:01	**02:53:30**	09:27:31	12:19:44	**02:52:13**	09:27:31	12:21:01	**02:53:30**	09:11:32	12:21:01	**03:09:29**	0	0	959
18	09:03:50	15:19:53	**06:16:03**	09:03:50	16:19:01	**07:15:11**	09:03:50	15:19:53	**06:16:03**	09:03:50	16:19:01	**07:15:11**	09:04:26	15:21:49	**06:17:23**	0	0	-3468
19	11:44:06	16:18:28	**04:34:22**	11:44:06	16:24:54	**04:40:48**	11:44:06	16:18:28	**04:34:22**	11:44:06	16:24:54	**04:40:48**	11:45:33	16:24:54	**04:39:21**	0	0	-87
20	10:11:54	11:03:22	**00:51:28**	10:11:54	11:01:49	**00:49:55**	10:11:54	11:03:22	**00:51:28**	10:11:54	11:01:49	**00:49:55**	10:17:41	11:01:49	**00:44:08**	0	0	-347
21	09:06:33	12:46:23	**03:39:50**	09:06:33	12:46:23	**03:39:50**	09:06:33	12:46:23	**03:39:50**	09:06:33	12:46:23	**03:39:50**	09:07:09	11:46:23	**02:39:14**	0	0	-3636
22	09:17:39	12:12:35	**02:54:56**	09:17:39	12:13:27	**02:55:48**	09:17:39	12:12:35	**02:54:56**	09:17:39	12:13:27	**02:55:48**	09:42:35	12:13:27	**02:30:52**	0	0	-1496
23	09:04:46	11:59:43	**02:54:57**	09:04:46	11:59:43	**02:54:57**	09:04:46	11:59:43	**02:54:57**	09:04:46	11:59:43	**02:54:57**	09:04:22	11:59:43	**02:55:21**	0	0	24
24	09:22:16	13:29:01	**04:06:45**	09:22:16	14:00:01	**04:37:45**	09:22:16	13:29:01	**04:06:45**	09:22:16	14:00:01	**04:37:45**	09:22:02	15:29:50	**06:07:48**	0	0	5403
25	09:02:45	11:27:28	**02:24:43**	09:02:45	11:30:55	**02:28:10**	09:02:45	11:27:28	**02:24:43**	09:02:45	11:30:55	**02:28:10**	09:01:11	11:30:55	**02:29:44**	0	0	94
26	09:02:33	13:13:22	**04:10:49**	09:02:33	13:11:30	**04:08:57**	09:05:44	13:13:22	**04:07:38**	09:06:39	13:11:30	**04:04:51**	09:06:39	13:11:30	**04:04:51**	-191	-246	-246
27	09:12:16	15:21:19	**06:09:03**	09:12:16	15:36:05	**06:23:49**	09:12:16	15:21:19	**06:09:03**	09:12:16	15:36:05	**06:23:49**	09:12:16	14:36:55	**05:24:39**	0	0	-3550
28	09:34:36	17:33:02	**07:58:26**	09:34:36	18:03:02	**08:28:26**	09:34:36	17:33:02	**07:58:26**	09:34:36	18:03:02	**08:28:26**	09:32:42	17:39:37	**08:06:55**	0	0	-1291
29	09:18:46	17:07:12	**07:48:26**	09:18:46	17:07:12	**07:48:26**	09:18:46	17:07:12	**07:48:26**	09:18:46	17:07:12	**07:48:26**	09:24:22	17:07:12	**07:42:50**	0	0	-336
30	09:15:37	14:46:48	**05:31:11**	09:15:37	14:50:42	**05:35:05**	09:15:37	14:46:48	**05:31:11**	09:15:37	14:50:42	**05:35:05**	09:17:13	14:52:15	**05:35:02**	0	0	-3
31	09:20:42	16:35:03	**07:14:21**	09:20:42	17:37:25	**08:16:43**	09:20:42	16:35:03	**07:14:21**	09:20:42	17:37:25	**08:16:43**	09:20:42	16:06:21	**06:45:39**	0	0	-5464
32	09:08:30	15:42:43	**06:34:13**	09:08:30	15:50:43	**06:42:13**	09:08:30	15:42:43	**06:34:13**	09:08:30	15:50:43	**06:42:13**	09:08:26	16:03:41	**06:55:15**	0	0	782
33	09:18:52	13:48:46	**04:29:54**	09:18:52	14:34:46	**05:15:54**	09:18:52	13:48:46	**04:29:54**	09:18:52	14:34:46	**05:15:54**	09:18:52	14:14:30	**04:55:38**	0	0	-1216
34	09:13:38	15:56:18	**06:42:40**	09:13:38	16:30:24	**07:16:46**	09:13:38	15:56:18	**06:42:40**	09:13:38	16:30:24	**07:16:46**	09:13:38	16:59:40	**07:46:02**	0	0	1756
35	09:20:44	15:22:58	**06:02:14**	09:20:44	15:52:58	**06:32:14**	09:20:44	15:22:58	**06:02:14**	09:20:44	15:52:58	**06:32:14**	09:20:44	15:09:44	**05:49:00**	0	0	-2594
36	09:22:40	15:04:05	**05:41:25**	09:22:40	15:04:05	**05:41:25**	09:22:40	15:04:05	**05:41:25**	09:22:40	15:04:05	**05:41:25**	09:22:40	14:36:05	**05:13:25**	0	0	-1680
37	09:33:59	14:24:49	**04:50:50**	09:33:59	14:24:49	**04:50:50**	09:33:59	14:24:49	**04:50:50**	09:33:59	14:24:49	**04:50:50**	09:32:06	14:24:49	**04:52:43**	0	0	113
38	09:04:17	14:15:50	**05:11:33**	09:04:17	14:45:50	**05:41:33**	09:04:17	14:15:50	**05:11:33**	09:04:17	14:45:50	**05:41:33**	09:03:53	14:26:34	**05:22:41**	0	0	-1132
39	09:05:45	15:46:53	**06:41:08**	09:05:45	17:46:53	**08:41:08**	09:06:57	15:46:53	**06:39:56**	09:06:57	17:46:53	**08:39:56**	09:05:45	15:27:57	**06:22:12**	-72	-72	-8336
40	09:29:23	17:14:45	**07:45:22**	09:29:23	17:15:31	**07:46:08**	09:29:23	17:14:45	**07:45:22**	09:29:23	17:15:31	**07:46:08**	09:31:49	16:28:52	**06:57:03**	0	0	-2945
41	09:06:06	14:57:42	**05:51:36**	09:06:06	16:41:28	**07:35:22**	09:06:06	14:57:42	**05:51:36**	09:06:06	16:41:28	**07:35:22**	09:15:06	15:44:36	**06:29:30**	0	0	-3952
42	09:47:32	16:33:07	**06:45:35**	09:47:32	16:55:53	**07:08:21**	09:47:32	16:33:07	**06:45:35**	09:47:32	16:55:53	**07:08:21**	09:47:32	15:45:34	**05:58:02**	0	0	-4219
43	09:04:58	14:20:25	**05:15:27**	09:04:58	14:53:29	**05:48:31**	09:04:58	14:20:25	**05:15:27**	09:04:58	14:53:29	**05:48:31**	09:20:50	14:53:35	**05:32:45**	0	0	-946
44	09:55:20	13:10:44	**03:15:24**	09:55:20	13:14:52	**03:19:32**	09:55:20	13:10:44	**03:15:24**	09:55:20	13:14:52	**03:19:32**	09:53:27	19:15:21	**09:21:54**	0	0	21742

Fig. 51: Result of the calculation from different scenarios

Figure 52 gives an overview about the trip time differences from scenario 1, scenario 2 and scenario 5. In all results, the routes include observed time periods from all buses used.

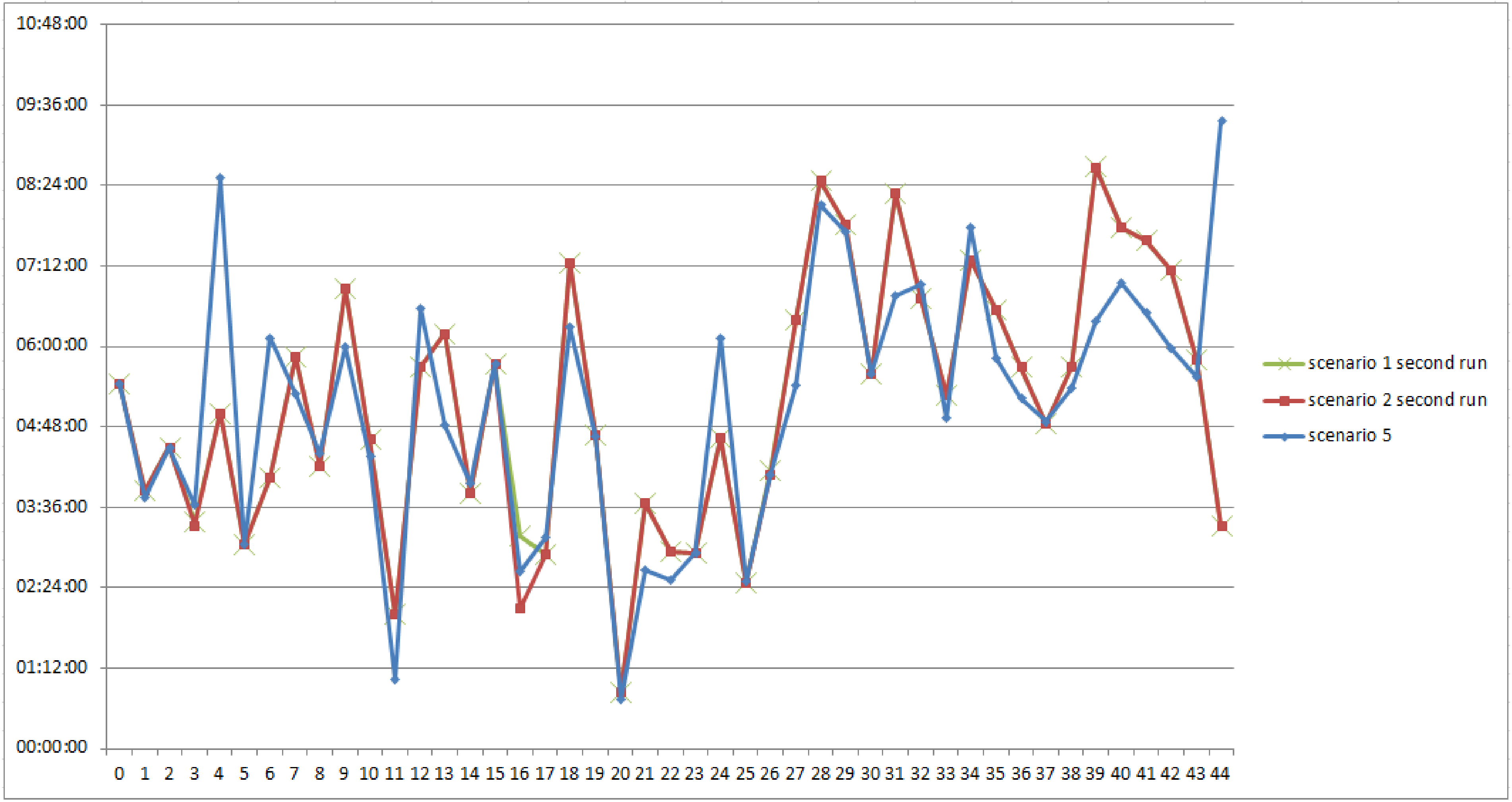

Fig. 52: Visualized trip time differences of scenario 1 and scenario 2 (both second run) as well as scenario 5

The following Figure 53 visualizes the trip times from scenario 1 and 2 from the first run, which means that these time periods are those that were calculated by the OpenTripPlanner and do not include observed stop_times. Generally, the trip length from the first run is faster than the length of the second run that includes observed time periods (see Figure 51).

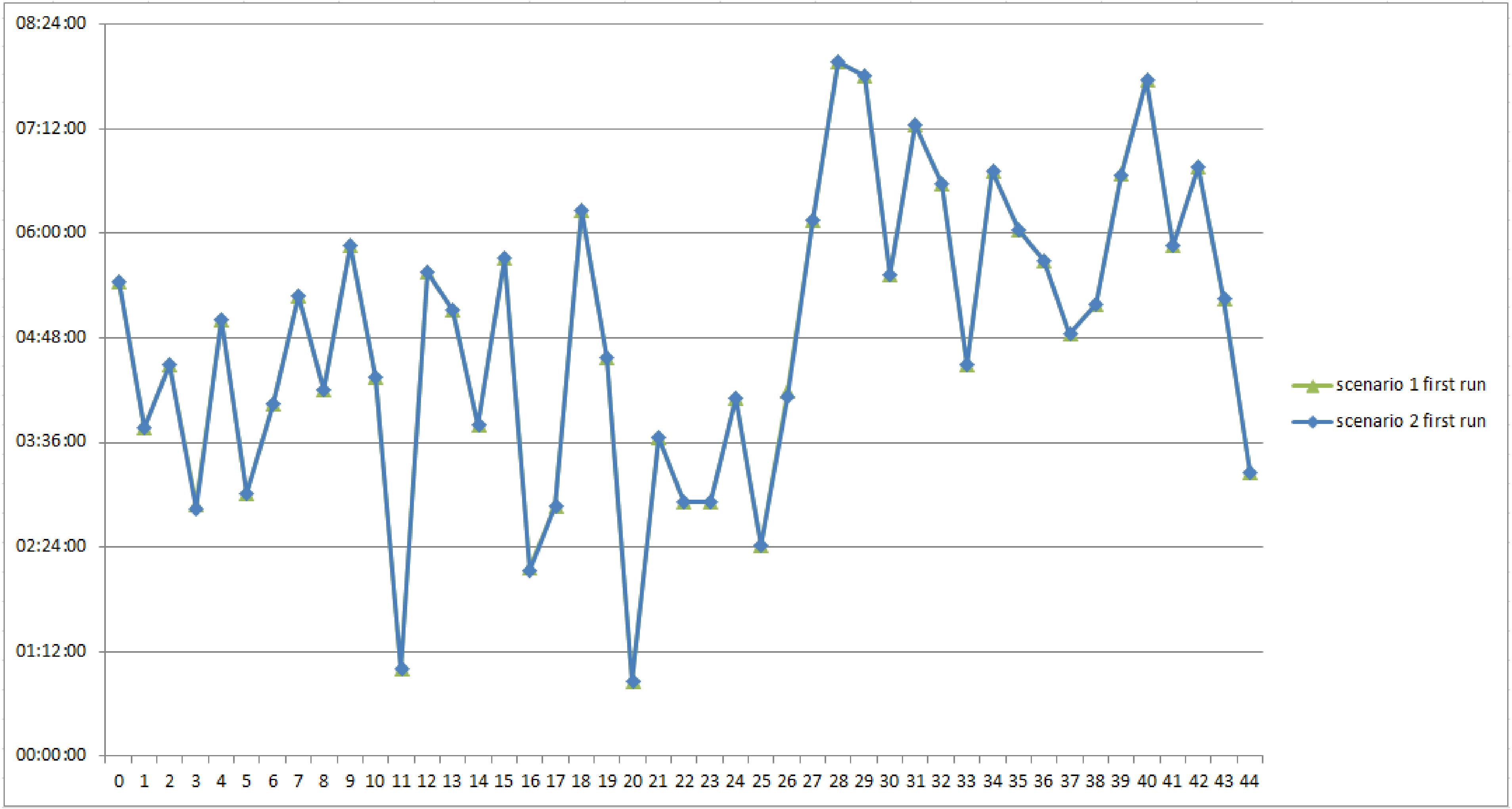

Fig. 53: Visualized trip times from scenario 1 and scenario 2 (both first run)

7. Discussion

This thesis mainly focused on the usage of Google's General Transit Feed Specification (GTFS) and the analysis of the available real-time data from the Netherlands. As the Netherlands is the (only) country in Europe that provides this type of real-time information to the public (since the end of January 2014), a detailed look at available public transit information was possible. In combination with the OpenTripPlanner, an extremely powerful route planning system, digital timetables as well as estimated future delays could be applied to the routing engine to perform evaluations about the effectiveness of real-time information.

The GTFS specification with its two types of real-time information ("tripUpdates" and "vehiclePositions") offers the base data for the analysis. The data structure can be seamlessly integrated with the static transit information and be implemented in a database. The spatial extension of PostgreSQL also enables the direct visualization of all collected information.

The continuously collected vehiclePositions for a certain time period made it possible to change the timetable of all affected trips to "observed" departure and arrival timestamps. That data has enabled both the comparison of estimated future delays for that period as well as the use of a modified timetable in the GTFS feed for route calculations of a pre-defined scenario (scenario 5). A similar research has already been done by Jariyasunant (2011).

In chapter 6.4, the delay analysis gives an interesting view of observed delays that were derived from the adapted timetable information out of vehiclePositions. Except the outliers due to its bad real-time data quality, the majority of delays occurred between 0 and 200 seconds, which is an acceptable range for passengers. Up to a range of +- 3σ from the normal distribution, delays don't increase to more than 600 seconds and don't decrease to less than -600 seconds.

Predicted future delays, which are published in the tripUpdate messages, were used for two purposes in this research: the comparison with observed delay information as well as the archived re-use for route calculations of a pre-defined scenario (scenario 2). In chapter 6.5, future delay information of tripUpdates was compared with departure times from vehiclePositions. During this analysis, a lack of future estimates for particular trips has been identified. On the one hand, real-time information of that kind is only available short before a certain trip starts, whereas on the other hand future delays during the trip are mostly considered being zero at the end of the trip. This lack of quality makes it almost impossible to provide passengers effective delay predictions.
To evaluate the effectiveness of real-time information to the passenger, different scenarios were defined that make use of the OpenTripPlanner (OTP). As customers make use of different types of route planning to get to their destination, variances of travel times with or without real-time information can be calculated and evaluated. In chapter 6.6, all different scenarios were simulated for random origins and destinations, and differences in travel times were compared.

8. Summary

This section summarizes essential results of this research. Furthermore, basic ideas for future work and improvements of the research are presented.

8.1 Conclusion

The main focus of this research was in the use of GTFS data feeds that allow developers to use public transit timetables as well as the spatial information behind it. To give an overview, this work was split into two parts. The initially defined part dealt with the implementation of a round trip recommendation for bicyclists in Austria. Because of the lack of open public transit information (Kurier 2013) in this country, a new main objective had to be defined. To summarize the new aim, real-time open public transit data from the Netherlands were analyzed regarding their effectiveness for passengers. To be able to perform this task, delay information had to be estimated and assessed.

During the research, a sample extract of a non-public GTFS feed from Österreichische Bundesbahnen (Austrian Federal Railways, ÖBB) in Austria could be acquired for testing purposes. Usually, ÖBB offers this type of data only to Google for its routing application Google Maps (DerStandard 2013). The extent of this sample feed was mainly in the Southern province of Austria, called Carinthia, whereas only railway data was included. The purpose was to demonstrate the broad field of application and the potential of publishing this type of information as open data to the public. The transit information was combined with an Open-Source route planner called OpenTripPlanner, which allows the calculation of a bike route in combination with public transit. This feature of multimodal routing, which isn't even available on Google Maps, could be a great opportunity for tourism agencies to boost the local bike tourism.

The second and main goal of this paper deals with the evaluation of real-time public transit information. As the only country in Europe, the Netherlands provides this kind of data as OpenData (OD) to the public. Up to now, only bus routes are included in this feed. Besides the static timetable information in the GTFS feed, two different types of real-time messages are continuously and up-to-date available: vehiclePositions and tripUpdates. The first ones include current positions of buses as well as information about the trip they are currently on. TripUpdates include predictions about future delays, which are included as additional information for passengers when planning a trip.

To assess the quality and effectiveness of real-time data, this information needed to be stored into a local database. For this, a PostgreSQL/PostGIS database has been created to import all necessary timetable information with stops and routes from the GTFS feed as well as the two real-time messages. Real-time feeds were recorded for almost one day, in time intervals of 30 seconds and below. The information could directly be linked to the GTFS feed tables. This used to be the base for evaluation purposes.

Using recorded information from vehiclePositions, it was possible to subsequently estimate and calculate the exact arrival and departure times of each bus in the network that is equipped with real-time modules. The observed arrival and departure times could be used to create a GTFS feed that contains the “real” timetables from the recorded time period. That modified feed was used in the scenario testing to calculate trips “as if they were optimal” for passengers.
Recorded tripUpdates were used to analyze future delay predictions, how they change over time for particular trips as well as the accuracy compared to observed arrival and departure times gained from vehiclePositions. Archived tripUpdates messages were also used in the scenario testing to act as a source of delay information when calculating a trip.

Last but not least, different scenarios were defined that should help to identify differences in travel times from an origin to a destination. These scenarios differ in using printed routes or smartphones with or without real-time information, as well as the route calculation using observed arrival and departure times to find the “optimal” route the passenger could have taken. As a conclusion to this simulation process, it was demonstrated that the initial travel period from the route planning process is mainly shorter than the observed travel period.

In a quick review, it is to mention that the real-time service in the Netherlands has been published in January 2014 and is still in a development phase. Up to now, only buses are capable of transmitting real-time positions, whereas in the testing time period of 7:00am to 8:00pm, just about 10 percent of all trips were covered (72.612 out of altogether 713.199 trips). Because of this fact, the modified GTFS feed that uses “real” arrival and departure times is just valid for trips that were really recorded. All other timetables remain the same.
The same issue is related to tripUpdates as well. During the collected time period, tripUpdates contained delay information for altogether 73.947 distinct trips, which is about 10 percent of all available ones. During the analysis phase, a lack of data quality in this message type has been identified (Figure 40 and 41). Predictions only become available shortly before that trip starts or after the trip has already started. According to the discussion in Transit Developers (2014a), it is very hard to predict future events and so delay information gets more and more inaccurate over time. The question here is, if it’s better not to give passengers the information about future delays with a high degree of uncertainty.

8.2 Future work

The current research of analyzing real-time public transit data from GTFS real-time feeds is a new approach of how to evaluate the quality of such information. Google was the pioneer of standardizing public transit data into a very slimmed and effective structure that can be used by any transit agency around the globe.

The advantages of this standard belong to all associated parties: Google is able to integrate public transit data from all over the world into its Maps, transit agencies profit from passengers that use their services and passengers profit from uncomplicated and more accurate trip planning across country borders. That's also the crucial point of that standard, which enables the combination of several feeds from adjoining countries to a single trip planning service. As the “de facto” global public transit standard, more and more agencies are going to publish their data in this format in the future.

Nowadays, as the extension to GTFS, GTFS real-time is published only by some single agencies around the world, because of its expensive infrastructure. Nevertheless, it offers a great opportunity for developers to create applications, especially for tourism purposes. The OpenTripPlanner offers a great OpenSource framework that is able to integrate GTFS feeds. The online community behind this project gives excellent feedback to all kind of concerns. For future research, this software could be extended by the described round trip recommendation in section 4.4, to enable tourists to plan a bicycle tour without the need of riding back the same way. Additionally, this functionality could be combined with public transit as well. Besides, another application could deal with park and ride suggestions that make use of real-time information to inform users about the fastest as well as cheapest way to reach the destination.
Since GTFS real-time is still in development, future services as well as the infrastructure behind it are going to be more accurate. The recorded real-time information sometimes shows bad data quality, where vehicles haven't followed the correct trip sequence or marked their timestamp of arrival far apart the timetable. TripUpdates also lack in delay prediction quality, which needs to be improved to guarantee more accurate predictions to the passengers.

Last but not least, real-time positions of vehicles could also be used to improve the timetables of the transit network. When collecting positions over time, mean arrival and departure times may be evaluated and used to adapt timetables, for instance in a yearly manner.

As a short summary, OpenData real-time services offer a modern way to provide up-to-date data to passengers as well as developers and enable the implementation of new applications as well as the opportunity of getting rid of traffic jams and unnecessary environmental pollution. The cornerstone to achieve this has already been set by Google.

9. References

ADFC, 2013: *Die ADFC-Radreiseanalyse 2013* [online]. Available from: http://www.adfc.de/radreiseanalyse/die-adfc-radreiseanalyse-2013 [last access 10.10.2013]

Afroz Ahmad, 2014: *Alternative to XML – Protocol Buffers* [online]. Available from: http://afrozahmad.hubpages.com/hub/protocolbuffers [last access 01.05.2014]

AnachB, 2013: *AnachB.at – neue Wege in der Vienna Region* [online]. Available from: http://www.anachb.at/ [last access 29.10.2013]

Bergfex, 2013: *Bergfex.at – Touren Österreich, Bike Kärnten* [online]. Available from: http://www.bergfex.at/sommer/kaernten/touren/bike/ [last acces 10.10.2013]

Bleyl, M., & Frommenwiler, T., 2012: *VDV 45x oder Google? Neue Schnittstellen für den Datenaustausch – Online PDF from itcs Seminar March 2012 Berlin* [online]. Available from: http://www.itcs-info.de/download/ITCS_Seminar_Mar_12/TO-05-VDV%2045x%20oder%20Google-Dr.%20Bleyl-ATRON.pdf [last access 29.04.2014]

Bronsveld, J., 2012: *Inter modal routing Parameters – What information is needed in the context of an inter modal trip advice and how can information be used?* [online]. Available from: http://essay.utwente.nl/61895/1/BronsveldJ_0087491_openbaar.pdf [last access 19.01.2014]

CEN, 2014: *CEN – European Committee for Standardization* [online]. Available from: http://www.cen.eu/cen/AboutUs/Pages/default.aspx [last access 18.01.2014]

Data KTN, 2014: *Open Government Data* [online]. Available from: http://data.ktn.gv.at/data/ [last access 06.01.2014]

Delling, D., et al, 2009: *Engineering Route Planning Algorithms. In: Algorithms of Large and Complex Networks, Lecture Notes in Computer Science Volume 5515, 2009, pp 117-139*

DerStandard, 2013: *ÖBB-Daten jetzt in Google Maps verfügbar* [online]. Available from: http://derstandard.at/1385171282859/OeBB-Daten-jetzt-in-Google-Maps-verfuegbar [last access 31.12.2013]

DFT, 2013: *Public Transport XML Standards* [online]. Available from: http://www.dft.gov.uk/public-transportdatastandards/ [last access 14.02.2014]

Dirksen, M., 2011: *Das OpenStreetMap-Projekt* [online]. Available from: http://www.informatik.uni-augsburg.de/lehrstuehle/dbis/db/lectures/ss11/mobileGIS/themen/Thema11_Ausarbeitung_Dirksen.pdf [last access 20.01.2014]

DrJTS, 2011: E*xtracting GTFS data using JEQL – Blog entry from October 7, 2011* [online]. Available from: http://lin-ear-th-inking.blogspot.com/2011/10/extracting-gtfs-data-using-jeql.html [last access 20.05.2014]

Eclipse, 2014: *About the Eclipse Foundation* [online]. Available from: http://www.eclipse.org/org/ [last access 28.05.2014]

Elias, B., & Hampe, M., 2003: *Kontextbezogene Kartengenerierung für Routing-Anwendungen. Technical Paper, Workshop Design kartenbasierter mobiler Dienste, Mensch und Computer* [online]. Available from: http://www.ikg.uni-hannover.de/fileadmin/ikg/staff/publications/sonstige_Beitraege/EliasHampe_Workshop2003.pdf [last access 19.01.2014]

Faludi, J., 2005: *A Call for Open-Source Public Transit Mapping* [online]. Available from: http://www.worldchanging.com/archives/002937.html [last access 17.01.2014]

FDGC, 2014: *National Spatial Data Infrastructure Framework Data – Framework Data Content Standard – Transportation: Transit* [online]. Available from: http://www.fgdc.gov/training/nsdi-training-program/materials/framwork-training-transportation-transit.pdf [last access 14.02.2014]

Garg, A., 2005: *Google Official Blog - Public transit via Google* [online]. Available from: http://googleblog.blogspot.co.at/2005/12/public-transit-via-google.html [last access 17.01.2014]

Geofabrik AUT, 2014: *Download OpenStreetMap data for this region: Austria* [online]. Available from: http://download.geofabrik.de/europe/austria.html [last access 15.03.2014]

Geofabrik NL, 2014: *Download OpenStreetMap data for this region: Netherlands* [online]. Available from: http://download.geofabrik.de/europe/netherlands.html [last access 13.04.2014]

GIP online, 2014: *GIP.at - GIP.at GIP.gv.at VAO* [online]. Available from: http://www.gip.gv.at/gipat-en.html [last access 20.01.2014]
GIP, 2010: GIP.at – Einheitlicher Verkehrsgraph für Österreich, Standardbeschreibung, Version 1.0, p. 13

GitHub GraphBuilder, 2014: *OpenTripPlanner GraphBuilder* [online]. Available from https://github.com/opentripplanner/OpenTripPlanner/wiki/GraphBuilder [last access 06.02.2014]

GitHub GraphStructure, 2014: *OpenTripPlanner GraphStructure* [online]. Available from: https://github.com/opentripplanner/OpenTripPlanner/wiki/GraphStructure [last access 05.02.2014]

GitHub Install, 2014: *Install – opentripplanner/OpenTripPlanner Wiki* [online]. Available from: https://github.com/opentripplanner/OpenTripPlanner/wiki/Install [last access 21.02.2014]

Github OTP, 2014a: *OpenTripPlanner Master Branch* [online]. Avalaiable from: https://github.com/opentripplanner/OpenTripPlanner/tree/master [last access 14.04.2014]

Github OTP, 2014b: *OpenTripPlanner Stable Branch* [online]. Available from: https://github.com/opentripplanner/OpenTripPlanner/tree/stable [last access 14.04.2014]

GitHub TwoMinutes, 2014: *TwoMinutes – Minimal Introduction to OTP* [online]. Available from: https://github.com/opentripplanner/OpenTripPlanner/wiki/TwoMinutes [last access 21.02.2014]

GitHub, 2014: *An open source multi-modal trip planner http://www.opentripplanner.org* [online]. Available from: https://github.com/opentripplanner/OpenTripPlanner [last access 21.02.2014]

Goodchild, M., 2007: *Citizens as sensors: the world of volunteered geography. In: GeoJournal, Volume 69, Issue 4, pp 211-221*

Google Developers, 2014a: *Protocol Buffers – What are Protocol Buffers?* [online]. Available from: https://developers.google.com/protocol-buffers/ [last access 28.02.2014]

Google Developers, 2014b: *Transit – gtfs-realtime.proto* [online]. Available from: https://developers.google.com/transit/gtfs-realtime/gtfs-realtime-proto [last access 28.02.2014]

Google Developers, 2014c: *Transit – Changes to GTFS-realtime* [online]. Available from: https://developers.google.com/transit/gtfs-realtime/changes [last access 29.04.2014]

Google Developers, 2014c: *Transit – Trip Updates* [online]. Available from: https://developers.google.com/transit/gtfs-realtime/trip-updates [last access 02.05.2014]

Google Developers, 2014d: *Transit – Vehicle Positions* [online]. Available from: https://developers.google.com/transit/gtfs-realtime/vehicle-positions [last access 02.05.2014]

Google Realtime, 2014: *Google Developers – What is GTFS-realtime?* [online]. Available from: https://developers.google.com/transit/gtfs-realtime/ [last access 24.01.2014]

Google Transit, 2014: *Google Maps – Transit* [online]. Available from: http://www.google.com/landing/transit/index.html [last access 17.01.2014]

GoogleTransitDataFeed, 2014: *PublicFeedsNonGTFS – Non-GTFS public transportation data feeds* [online]. Available from: http://code.google.com/p/googletransitdatafeed/wiki/PublicFeedsNonGTFS [last access 25.04.2014]

GTFS Data Exchange, 2014: *Website for GTFS data exchange* [online]. Available from: http://www.gtfs-data-exchange.com, last access 17.01.2014

GTFSRT, 2014: *GTFS-realtime – Google Groups* [online]. Available from: https://groups.google.com/forum/#!forum/gtfs-realtime [last access 21.02.2014]

Heitzmann, R., 2014: *Bob Heitzman's Google Site – XLS Tools for Google Transit Feed Specification (GTFS)*
Hickman, M., & Wilson, N., 1995: Passenger travel time and path choice implications of real-time transit information. Trans. Research C, 3(4):211–226

Hillsman, E., & Barbeau, S., 2011*: Enabling Cost-Effective Multimodal Trip Planners through Open Transit Data (No. USF 21177926)* [online]. Available from: http://www.dot.state.fl.us/research-center/Completed_Proj/Summary_PTO/FDOT_BDK85_977-20_rpt.pdf [last access: 19.01.2014]

Hochmair, H., & Navratil, G., 2008: *Computation of scenic routes in street networks. In: Geospatial Crossroads@ GI_Forum'08: Proceedings of the Geoinformatics Forum Salzburg (pp. 124-133)* [online]. Available from: ftp://128.130.78.175/navratil/HochmairNavratilAGIT08.pdf [last access 19.01.2014]

Hochmair, H., & Rinner, C., 2005: *Investigating the need for eliminatory constraints in the user interface of bicycle route planners. In: Spatial Information Theory (pp. 49-66). Springer Berlin Heidelberg.*

Hochmair, H., Zielstra, D., & Neis, P., 2013: *Assessing the Completeness of Bicycle Trail and Designated Lane Features in Open-StreetMap for the United States. In: Transactions in GIS.*

Hughes, J., 2007: *General Transit Feed Spec Changes* [online]. Available from: https://groups.google.com/forum/#!msg/gtfs-changes/C5dgsKGkpDA/kyxN1DCS-dQJ [last access 17.01.2014]

Jariyasunant, J., et al, (2011): *Mobile transit trip planning with real-time data* [online]. Available from: http://escholarship.org/uc/item/51t364vz [last access 09.09.2014]

Java, 2014: *What is Java technology and why do I need it?* [online]. Available from: http://www.java.com/en/download/faq/whatis_java.xml [last access 28.05.2014]

JEQL, 2014: *JEQL Query Language* [online]. Available from: http://tsusiatsoftware.net/jeql/main.html [last access 20.05.2014]

Joest, M., & Stille, W., 2002: *A user-aware tour proposal framework using a hybrid optimization approach. In: Proceedings of the 10th ACM international symposium on Advances in geographic information systems (pp. 81-87). ACM.*

Kaernten, 2013: Kaernten – Lust am Leben, Interactive Map [online]. Available from: http://alpregio.outdooractive.com/ar-kaernten/de/alpregio.jsp#activ=Rad&tab=ToursTab [last access 10.10.2013]

Kaufman, S., & Wagner, R., 2012: *Getting Started with Open Data: A Guide for Transportation Agencies* [online]. Available from: http://wagner.nyu.edu/files/rudincenter/opendata.pdf [last access 17.01.2014]

Kizoom, N., & Miller, P., 2008: *A Transmodel based XML schema for the Google Transit Feed Specification – With a GTFS / Transmodel comparison* [online]. Available from: http://www.dft.gov.uk/transmodel/schema/doc/GoogleTransit/TransmodelForGoogle-09.pdf [last access 14.02.2014]

Komoot, 2013: *Komoot – Tourenvorschläge, Tour planen, Apps* [online]. Available from: http://www.komoot.de/suggest/#du=10800 [last access 08.10.2013]

Kurier, 2013: *ÖBB: Kein Fahrplan für App-Entwickler* [online]. Available from: http://kurier.at/lebensart/technik/oebb-kein-fahrplan-fuer-app-entwickler/41.545.631 [last access 31.12.2013]

Linz AG Linien, 2014: *Elektronische Fahrplanauskunft der LINZ AG LINIEN - Sollzeit* [online]. Available from: http://www.data.gv.at/datensatz/?id=909db36e-adec-485a-858d-c4f74a386305 [last access 06.01.2014]

Litman, T., 2011: *Introduction to Multi-Modal Transportation Planning. Victoria Transport Policy Institute, 15* [online]. Available from: http://www.vtpi.org/multimodal_planning.pdf [last access 06.01.2014]

Lyoen, C., Van den Ven, T., Kim Lim, K., 2010: *ITS Action Plan – D5 – Consultation Report – Study regarding guaranteed access to traffic and travel data and free provision of universal traffic information* [online]. Available from: http://ec.europa.eu/transport/themes/its/studies/doc/2010_12-consultation-report-study-data-access-free-safety-traffic-information.pdf [last access 18.01.2014]

Mandl-Mair, I., 2014: *GIP.at – Einheitlicher Verkehrsgraph für Österreich* [online]. Available from: http://www.kagis.ktn.gv.at/194378_DE-Dateien-GIPVortragAGIT.pdf [last access 15.01.2014]

Maps, 2013: *Google Maps* [online]. Available from: https://maps.google.at/ [last access 29.10.2013]

McHugh, B., 2011: *The OpenTripPlanner Project – Final Report* [online]. Available from: http://portlandtransport.com/documents/OTP%20Final%20Report%20-%20Metro%202009-2011%20RTO%20Grant.pdf [last access 20.01.2014]

McHugh, B., 2013: *Pioneering Open Data Standards: The GTFS Story* [online]. Available from: http://beyondtransparency.org/chapters/part-2/pioneering-open-data-standards-the-gtfs-story/ [last access 17.01.2014]

Moccia, L., Cordeau, J. F., Laporte, G., Ropke, S., & Valentini, M. P., 2008: *Modeling and solving a multimodal routing problem with timetables and time windows. CIRRELT* [online]. Available from: https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2008-28.pdf [last access 10.10.2013]

Molin, H., 2012: *Access to Public Data for Digital Road Maps – The intermodal Transport Reference System for Austria (Graphs Integration Platform – GIP)* [online]. Available from: http://ec.europa.eu/transport/themes/its/events/doc/2011_03_29/2011_03_29-42-gip-austria.pdf [last access 14.02.2014]

Muenster, 2014: *Muensterlandradweg.de – Radtouren-Vorschlaege* [online]. Available from: http://www.muensterlandradweg.de/radtouren-vorschlaege [last access 15.01.2014]

Navteq, 2014: *Navmart – Navteq data for spatial technology* [online]. Available from: http://www.navmart.com/navteq_navstreets.php [last access 20.01.2014]

ODF, 2013*: Open Data Foundation – supporting the vision of open and accessible data up and down the information chain* [online]. Available from: http://www.opendatafoundation.org/ [last access 22.10.2013]

Open Definition, 2005: *Open Definition - The Open Definition in Your Language* [online]. Available from: http://opendefinition.org/ [last access 8.10.2013]

OSD, 2013: *OpenSource Definition – The Open Source Definition | Open Source Initiative* [online]. Available from: http://opensource.org/osd [last access 22.10.2013]

OSM Mapnik, 2014: *Mapnik* [online]. Available from: http://wiki.openstreetmap.org/wiki/Mapnik [last access 20.01.2014]

OSM Slippy, 2014: *Slippy map* [online]. Available from: http://wiki.openstreetmap.org/wiki/Slippy_Map [last access 20.01.2014]

OSM Stats, 2014: *OpenStreetMap stats report run at 2014-01-20 00:00:12 + 0000* [online]. Available from: http://www.openstreetmap.org/stats/data_stats.html [last access 20.01.2014]

OSM, 2013: *OpenStreetMap – Die freie Wiki-Weltkarte* [online]. Available from: http://www.openstreetmap.org/#map=5/51.500/-0.100 [last access 08.10.2013]

Osmosis, 2014: *Osmosis/Replication* [online]. Available from: http://wiki.openstreetmap.org/wiki/Osmosis/Replication [last access 20.01.2014]

OTP Developers, 2014: *OpenTripPlanner Developers – Google Groups* [online]. Available from: https://groups.google.com/forum/#!forum/opentripplanner-dev [last access 21.02.2014]

OTP NL, 2014: *Opentripplanner Netherlands* [online]. Available from: http://opentripplanner.nl/index.html [last access 15.01.2014]

OTP Tampa, 2014: *Opentripplanner Tampa* [online]. Available from: http://opentripplanner.usf.edu/#/ [last access 15.01.2014]

OTP Users, 2014a: *OpenTripPlanner Users – Google Groups* [online]. Available from: https://groups.google.com/forum/#!forum/opentripplanner-users [last access 21.02.2014]

OTP Users, 2014b: *OpenTripPlanner Users – Topic "GTFS Realtime in OTP (master branch)* [online]. Available from: https://groups.google.com/forum/#!msg/opentripplanner-users/F1MTMQW1B1Y/BbTU61t0uKUJ [last access 03.03.2014]
OTP Users, 2014d: *OpenTripPlanner Users - How to use an elevation file in master branch?* [online]. Available from: https://groups.google.com/forum/#!topic/opentripplanner-users/kwE_DCOO8pw [last acccess 25.05.2014]

OTP Wiki, 2013: *OpenStreetMap Wiki* [online]. Available from: http://wiki.openstreetmap.org/wiki/OpenTripPlanner [last access 10.10.2013]

OTP, 2013: *OpenTripPlanner (OTP) – the leading open source platform for multimodal trip itinerary planning and network analysis* [online]. Available from: http://opentripplanner.com/ [last access 10.10.2013]

Outdooractive, 2014: *outdooractive.com – Touren finden und planen* [online]. Available from: http://alpregio.outdooractive.com/ar-kaernten-mitte/de/alpregio.jsp#tab=WelcomeTab [last access 15.01.2014]

OVapi, 2014: *Directory of GTFS-realtime information from Netherlands* [online]. Available from: http://gtfs.ovapi.nl/nl/ [last access 12.02.2014]

PT Wiki, 2014: *Public Transport* [online]. Available from: http://en.wikipedia.org/wiki/Public_transport [last access 17.01.2014]

Radtouren, 2014: *Radtouren in Oesterreich* [online]. Available from: http://www.radtouren.at/radtouren.html [last access 15.01.2014]

railML, 2013a: *Feasability Study – UIC RailTopoModel and data exchange format, Zürich/Paris* [online]. Available from: http://www.railml.org/tl_files/railML.org/documents/science/201213_UIC_RailTopoModel_DraftDec13.pdf [last access 25.02.2014]

railML, 2013b: *Final Report – Feasability Study – UIC RailTopoModel and data exchange format, Zürich/Paris* [online]. Available from: http://railml.org/tl_files/railML.org/documents/science/270913_trafIT_FinalReportFeasibilityStudyRailTopoModel.pdf [last access 21.08.2014]

RailML, 2014: *Need for a common exchange format* [online]. Available from: http://www.railml.org/index.php/railml3-development.html [last access 25.02.2014]

Ramthun, R., 2012: *Offene Geodaten durch OpenStreetMap. In: Saarbrücker Schriften zur Informationswissenschaft, Open Initiatives: Offenheit in der Digitalen Welt und Wissenschaft, pp 159-184, Saarland University Press*

Raschke, 2011: *Open Standards for AVL and other real-time transit data* [online]. Available from: http://transport.kurtraschke.com/2011/04/open-standards-for-avl [last access 14.02.2014]

Reed, L. T., 2013: *Real-time transit passenger information: a case study in standards development. Georgia Institute of Technology* [online]. Available from: http://smartech.gatech.edu/handle/1853/50218 [last access 25.01.2014]

Roush, W., 2012: *Google Transit: How (and Why) the Search Giant is Remapping Public Transportation* [online]. Available from: http://www.xconomy.com/san-francisco/2012/02/21/google-transit-a-search-giant-remaps-public-transportation/5/ [last access 11.02.2014]

Route NRW, 2014: *Radroutenplaner NRW – Der Routenplaner fuer das Fahrrad* [online]. Available from: http://radservice.radroutenplaner.nrw.de/rrp/nrw/cgi?lang=DE [last access 15.01.2014]

Scriptol, 2014: *How to use Protocol Buffers* [online]. Available from: http://www.scriptol.com/programming/protocol-buffers-tutorial.php [last access 29.05.2014]

Soares, I., & Martins, P., 2013: *Pubic Transport Standardization – A Contribution to the State of the Art Review* [online]. Available from: http://labs.integra-travel.eu/files/paper_standards_WCTR.pdf [last acccess 14.02.2014]

Stanford, 2014: *Introduction to A* - From Amit's Thoughts on Pathfinding* [online]. Available from: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html [last access 19.01.2014]

STK, 2012: *Wie schwer es fallen kann, Fahrplaene zu oeffnen* [online]. Available from: http://stefan.bloggt.es/2012/09/wie-schwer-es-fallen-kann-fahrplaene-zu-oeffnen/ [last access 11.02.2012]

Stroemer, K., 2006: *Konzeption eines flexiblen Routingalgorithmus auf Basis einer empirischen Untersuchung zum Verhalten von Radfahrern beim Planen einer Farradtour. In STROBL, J., BLASCHKE, T. & G. GRIESEBNER. In: Angewandte Geoinformatik (pp. 651-655)* [online]. Available from: http://www.agit.at/php_files/myAGIT/papers/2006/5524.pdf [last access 15.01.2014]

Szabo, R., et al, 2013: *Measurements of a real-time transit feed service architecture for mobile participatory sensing*. In Wireless Days (WD), 2013 IFIP (pp. 1-4). IEEE, Valencia. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6686542&tag=1 [last access 09.09.2014]

Teleatlas, 2013: *Teleatlas – Detaillierte Stadtpläne für tausende Städte* [online]. Available from: http://www.teleatlas.at/ [last access 01.11.2013]

Teleatlas, 2014: *TELEDAT: TeleAtlas-Daten, Produktbeschreibung* [online]. Available from: http://files.be.ch/bve/agi/geoportal/geo/lpi/TELEDAT_2012_1_LANG_DE.pdf [last access 20.01.2014]

TFGM, 2014: *What is public transport?* [online]. Available from: http://www.tfgm.com/Corporate/Documents/Miscellaneous/work.pdf [last access 17.01.2014]

TourIS, 2013: *TourIS Kärnten – Tourismus-Informations-System der Kärnten Werbung* [online]. Available from: http://www.touris.kaernten.at/?arid=5639&jid=SO&siid=1247 [last access 10.10.2013]

Transit Cities, 2014: *Google Maps – Transit – Cities Covered* [online]. Available from: http://www.google.com/landing/transit/cities/index.html [last access 17.01.2014]

Transit Developers, 2014a: *GTFS Feeds for The Netherlands, in: Google Groups, Transit Developers* [online]. Available from: https://groups.google.com/forum/#!msg/transit-developers/MbGRNM9keJ8/Z9ExR65YZHsJ [last access 14.02.2014]

Transit Developers, 2014b: *TransitDevelopers – Google Groups* [online]. Available from: https://groups.google.com/forum/#!forum/transit-developers [last access 21.02.2014]

Transitfeeds, 2014: *Transitfeeds Europe* [online]. Available from: http://transitfeeds.com/l/60-europe [last access 11.02.2014]

TriMet, 2014: *TRIMET Trip Planner* [online]. Available from: http://maps.trimet.org/ [last access 15.01.2014]

UITP, 2014: *International Association of Public Transport – Connecting the world of public transport* [online]. Available from: http://www.uitp.org/Public-Transport/why-public-transport/ [last access 17.01.2014]

USGS, 2013: *USGS – Global Data Explorer (Powered by GeoBrain)* [online]. Available from: http://gdex.cr.usgs.gov/gdex/ [last access 30.10.2013]

VA Oesterreich, 2014: *Verkehrsauskunft Oesterreich – Beta Test* [online]. Available from: http://beta.verkehrsauskunft.at/bin/query.exe/dn [last access 15.01.2014]

VDV Schriften, 2013: *ÖPNV Datenmodell 5.0 – "Schnittstellen-Initiative" – VDV-Standardschnittstelle Liniennetz/Fahrplan, Version 1.5* [online]. Available from: http://www.vdv.de/452--sdsv15.pdfx?forced=true [last access 29.04.2014]

VDV, 2014a: *VDV Projects - CEN TS 15531 Service Interface for Real time Information (SIRI)* [online]. Available from: http://mitglieder.vdv.de/en/wir_ueber_uns/vdv_projekte/siri.html [last access 27.02.2014]

VDV, 2014b: VDV Projects – Integration Interface for Automatic Vehicle Management Systems [online]. Available from: http://mitglieder.vdv.de/en/wir_ueber_uns/vdv_projekte/istdaten_schnittstellen.html?pe_id=50 [last access 27.02.2014]

VDV, 2014c: VDV-Standardschnittstelle Liniennetz/Fahrplan (German Translation of VDV 452) [online]. Available from: http://www.vdv.de/452--sdsv15.pdfx?forced=true [last access 15.05.2014]

VDV, 2014d: VDV Standard Interface Network / Timetable [online]. Available from: http://mitglieder.vdv.de/module/layout_upload/452_sesv14.pdf [last access 15.05.2014]

Jariyasunant, J., Kerkez, B., Sengupta, R., Glaser, S., & Bayen, A., 2011: *Mobile transit trip planning with real-time data. University of California Transportation Center* [online]. Available from: http://escholarship.org/uc/item/51t364vz.pdf [last access 30.04.2014]

Vogella, 2014: JAXB – Tutorial [online]. Available from: http://www.vogella.com/tutorials/JAXB/article.html [last access 10.09.2014]

Weyrer, T., 2013: *Ready.Set.Access – Development of an Intermodal Door2Door Routing Application for People with Physical Impairments. Master Thesis, Carinthia University of Applied Sciences, Villach*

WIEN, 2013: *AnachB.at - New journey planning apps for smart phones* [online]. Available from: http://www.wien.gv.at/english/transportation-urbanplanning/routeplanning-app.html [last access 29.10.2013]

Wiener Linien, 2014: *Wiener Linien – Echtzeitdaten* [online]. Available from: http://data.gv.at/datensatz/?id=add66f20-d033-4eee-b9a0-47019828e698 [last access 06.01.2014]

WIKI Bicycle, 2013: *Bicycle – OpenStreetMap Wiki* [online]. Available from: http://wiki.openstreetmap.org/wiki/Bicycle [last access 30.10.2013]

Wong, J. C., 2013: *Use of the general transit feed specification (GTFS) in transit performance measurement* (Doctoral dissertation, Georgia Institute of Technology).

Zipf, A., & Jöst, M., 2006: *Implementing adaptive mobile GI services based on ontologies: Examples from pedestrian navigation support. Computers, Environment and Urban Systems, Volume 30, Issue 6, pp 784-798, University of Applied Sciences FH Mainz, Mainz, Germany*

Zukunft Mobilitaet, 2014: *Artikel zum Thema GTFS Data Exchange - Was passiert eigentlich, wenn das Auto den ÖPNV als bessere Alternative vorschlägt?* [online]. Available from: http://www.zukunft-mobilitaet.net/thema/gtfs-data-exchange/ [last access 03.03.2014]

10. List of Figures

11. List of Tables

12. Appendix

JQL Code to convert a GTFS feed into shapefiles:

```
CSVReader tshape hasColNames: file: "C:\\shp
ovapi\\test\\ovapi\\shapes.txt";
CSVReader ttrip hasColNames: file: "C:\\shp ovapi\\test\\ovapi\\trips.txt";
CSVReader troutes hasColNames: file: "C:\\shp
ovapi\\test\\ovapi\\routes.txt";
Mem troutes;
crs_WSG84 = "EPSG:4326";
trips = select distinct route_id, trip_id, realtime_trip_id, trip_headsign,
shape_id
    from ttrip;
tpt = select shape_id, Val.toInt(shape_pt_sequence) seq,
        Geom.createPoint(Val.toDouble(shape_pt_lon),
                                 Val.toDouble(shape_pt_lat) ) pt
    from tshape
    order by shape_id, seq;
tline = select shape_id,
     CRS.project(Geom.toMulti(geomConnect(pt)), crs_WSG84) geom
    from tpt
    group by shape_id;
trouteLines = select r.route_id, r.agency_id,
    r.route_short_name, r.route_long_name,
    r.route_type, r.route_url,
    trip_id, realtime_trip_id, trip_headsign, l.*
        from tline l
        join trips t on t.shape_id == l.shape_id
        join troutes r on r.route_id == t.route_id;
ShapefileWriter trouteLines file: "C:\\shp ovapi\\test\\gtfs_trips.shp";
CSVReader tstopsRaw hasColNames: file: "C:\\shp
ovapi\\test\\ovapi\\stops.txt";
tstop = select stop_id,stop_code,stop_name,
        CRS.project(Geom.createPoint(
        Val.toDouble(stop_lon),
        Val.toDouble(stop_lat) ), crs_WSG84) pt,
        zone_id
        from tstopsRaw;
ShapefileWriter tstop file: "C:\\shp ovapi\\test\\gtfs_stops.shp";
```

graph-builder.xml for OTP stable:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd">
    <bean id="gtfsBuilder" class="org.opentripplanner.graph_builder.impl.GtfsGraphBuilderImpl">
        <property name="gtfsBundles">
            <bean id="gtfsBundles" class="org.opentripplanner.graph_builder.model.GtfsBundles">
                <property name="bundles">
                    <list>
                        <bean class="org.opentripplanner.graph_builder.model.GtfsBundle">
                            <!-- uses a source from a homepage -->
                            <!--  <property name="url" value="http://developer1.trimet.org/schedule/gtfs.zip" />   -->
                            <!-- uses a source from a local path -->
                            <property name="path" value="/var/otp/downloads/gtfs-austria.zip" />
                            <!-- By default, bikes may only be taken along on transit trips if the GTFS data allows them
                                 to be. If the GTFS data doesn't contain appropriate data, but the actual transit trips
                                 do allows bikes to be taken along, a defaultBikesAllowed property may be specified to
                                 allow bikes. -->
                            <property name="defaultBikesAllowed" value="true" />
                        </bean>
                    </list>
                </property>
            </bean>
        </property>
        <!-- GTFS-rt ALERTS -->
        <property name="gtfsGraphBuilders">
            <list>
                <bean class="org.opentripplanner.graph_builder.impl.transit_index.TransitIndexBuilder" />
            </list>
        </property>
    </bean>
    <bean id="osmBuilder" class="org.opentripplanner.graph_builder.impl.osm.OpenStreetMapGraphBuilderImpl">
        <!-- Use an OSM provider that reads a file -->
        <property name="provider">
            <bean class="org.opentripplanner.openstreetmap.impl.AnyFileBasedOpenStreetMapProviderImpl">
```

```
                <property name="path" value="/var/otp/downloads/austria.osm.pbf" />
            </bean>
        </property>
        <property name="defaultWayPropertySetSource">
            <bean class="org.opentripplanner.graph_builder.impl.osm.DefaultWayPropertySetSource" />
        </property>
        <!-- custom unamed street namer -->
        <property name="customNamer">
            <bean class="org.opentripplanner.graph_builder.impl.osm.PortlandCustomNamer" />
        </property>
    </bean>
    <bean id="nedBuilder" class="org.opentripplanner.graph_builder.impl.ned.NEDGraphBuilderImpl">
        <property name="gridCoverageFactory">
                <bean class="org.opentripplanner.graph_builder.impl.ned.GeotiffGridCoverageFactoryImpl">
                    <property name="path" value="/var/otp/cache/ned/ASCIITo_DHM_1.tif" />
                </bean>
        </property>
    </bean>
    <bean id="transitStreetLink" class="org.opentripplanner.graph_builder.impl.TransitToStreetNetworkGraphBuilderImpl"/>
    <bean id="checkGeometry" class="org.opentripplanner.graph_builder.impl.CheckGeometryGraphBuilderImpl" />
    <bean id="graphBuilderTask" class="org.opentripplanner.graph_builder.GraphBuilderTask">
        <property name="path" value="/otp" />
        <property name="graphBuilders">
            <list>
                <ref bean="gtfsBuilder" />
                <ref bean="osmBuilder" />
                <ref bean="nedBuilder" />
                <ref bean="checkGeometry" />
                <ref bean="transitStreetLink" />
            </list>
        </property>
    </bean>
</beans>
```

Frequency table of delay distribution (-1800s < x < 1800s):

delay_groups delay (Binned)

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	2 -1799 - -1700	120	.0	.0	.0
	3 -1699 - -1600	1153	.1	.1	.1
	4 -1599 - -1500	123	.0	.0	.1
	5 -1499 - -1400	114	.0	.0	.1
	6 -1399 - -1300	146	.0	.0	.1
	7 -1299 - -1200	186	.0	.0	.1
	8 -1199 - -1100	173	.0	.0	.1
	9 -1099 - -1000	191	.0	.0	.2
	10 -999 - -900	253	.0	.0	.2
	11 -899 - -800	304	.0	.0	.2
	12 -799 - -700	447	.0	.0	.2
	13 -699 - -600	591	.0	.0	.3
	14 -599 - -500	1020	.1	.1	.3
	15 -499 - -400	2167	.2	.2	.5
	16 -399 - -300	5097	.4	.4	.9
	17 -299 - -200	16452	1.2	1.2	2.0
	18 -199 - -100	62721	4.4	4.4	6.4
	19 -99 - 0	232176	16.4	16.4	22.8
	20 1 - 100	490211	34.5	34.5	57.3
	21 101 - 200	310406	21.9	21.9	79.2
	22 201 - 300	146456	10.3	10.3	89.5
	23 301 - 400	67535	4.8	4.8	94.3
	24 401 - 500	33389	2.4	2.4	96.6
	25 501 - 600	16924	1.2	1.2	97.8
	26 601 - 700	9789	.7	.7	98.5
	27 701 - 800	6234	.4	.4	99.0
	28 801 - 900	3966	.3	.3	99.2
	29 901 - 1000	2680	.2	.2	99.4
	30 1001 - 1100	1794	.1	.1	99.6
	31 1101 - 1200	1390	.1	.1	99.7
	32 1201 - 1300	1063	.1	.1	99.7
	33 1301 - 1400	723	.1	.1	99.8
	34 1401 - 1500	691	.0	.0	99.8
	35 1501 - 1600	565	.0	.0	99.9
	36 1601 - 1700	1511	.1	.1	100.0
	37 1701+	292	.0	.0	100.0
	Total	1419053	100.0	100.0	