Quality Assessment of Open Realtime Data for Public Transportation in the Netherlands

Daniel Steiner[1], Hartwig Hochmair[2] and Gernot Paulus[1]

[1]Carinthia University of Applied Sciences, Villach/Austria · daniel.steiner@edu.fh-kaernten.ac.at
[2]University of Florida, Fort Lauderdale/USA

Full paper double blind review

Abstract

Modern technologies in the public transportation sector offer a variety of opportunities to advance solutions for every-day use, such as trip planning. Oftentimes, buses and metro lines do not adhere to their schedules, e.g. due to traffic congestion or overcrowding. In such cases, realtime information on the transit system may be used to find faster alternative routes. This paper analyses the quality of a realtime data set that is the first of its kind in Europe, since it covers a whole country, i.e. the Netherlands. The data are published in the General Transit Feed Specification (GTFS)-realtime format as a live-feed. These interoperable feeds, if shared with the public by transit agencies, can be used by developers to write customized applications for public transit users, e.g. through integration with the OpenTripPlanner (OTP) routing platform.

1 Introduction

Transit users have certain demands on public transportation systems, including fast and reliable on-schedule services (EL-GENEIDY et al. 2010), which cannot always be provided. One common problem in this context is, for example, the phenomenon of bus bunching, in which multiple buses on the same line arrive at a stop concurrently, followed by no subsequent buses for a significant amount of time (DESSOUKY et al. 2003). Given such potentially low system reliability values, realtime information can be used to compute faster routes and decrease waiting time (ROSSETTI & TURITTO 1998). To obtain realtime system information, many transit agencies employ automatic vehicle location (AVL) and automatic passenger counter (APC) technologies on their buses. JARIYASUNANT et al. (2012) assessed the benefits of realtime information on mobile devices for transit trip planning, when compared to routes based on a static schedule. The use of realtime data slightly reduced the median travel time prediction error from 14.9% to 11.7%.

Providing realtime estimated departure times and live vehicle positions as a realtime feed requires significant resources for system operation and management. This issue deters many agencies from allocating such realtime feeds for open access (ROUSH 2012). Several transit agencies in the US already provide such live feeds, however, realtime data have so far only been available for two European regions. OVapi (accessible at http://gtfs.ovapi.nl/) introduced the access to the realtime feeds for buses from different transit agencies in the Netherlands at the end of January 2014, using the GTFS (General Transit Feed Specifi-

cation) realtime format. This format can be integrated with the open source software OpenTripPlanner (OTP). EMT, the public bus company of Madrid, representing the second region, provides realtime bus stop delays in a different open data format. Since OVapi delivers data in the widely accepted GTFS-realtime feed format, and their data covers a larger geographic area than Madrid, this paper focuses on the data quality evaluation of the OVapi real time information. Among the several commonly used measures of geodata quality, this paper analyses data completeness and temporal accuracy of the realtime feeds. OTP will be used to illustrate the effect of GTFS-realtime information on selected computed bus routes.

2 Realtime Public Transit Data

2.1 Availability

In 2011, when the GTFS-realtime format was established, several cities in the US and Europe (i.e. Boston, Portland, San Diego, San Francisco, Madrid, and Turin) were invited by Google to deliver their realtime data to Google Maps for integration into Google's multimodal route planner (ROUSH 2012). For Turin these realtime feeds were not directly shared with the public in a downloadable form, whereas the public bus company EMT Madrid provides realtime bus stop delays as open data. For other European cities, such as Linz or Vienna, tracked realtime vehicle positions and estimated vehicle departure and arrival times are integrated in the proprietary routing engine of the transit agencies. Trip computations are also published as a Web service that can be called from within other applications through the API. However, it is not (yet) possible to download the realtime information of the entire network at a given time, which would be necessary for building customized routing applications. Thus, in Europe only OVapi provides Open Access to GTFS-realtime data so far, whereas it is already provided by several US transit agencies, such as BART (Oakland, CA), TriMet (Oregon, WA), or MARTA BUS (Atlanta, GA).

2.2 Public Transit Data Standards

In addition to GTFS-realtime, which can already be considered a worldwide de-facto industry standard, public transit standards are also defined at the European or national (e.g. Germany) level. These include SIRI for the EU, NextBus for the US and Canada, or VDV Interfaces 353/354 for Germany. In the original, non-realtime GTFS standard each static GTFS feed consists of a set of comma separated value (CSV) files, which are compressed in a ZIP package and linked with each other through relation IDs. The GTFS-realtime extension currently supports three types of realtime information:

- **VehiclePositions**: information about the vehicles including current location.
- **Trip updates**: delays, cancellations and changed routes.
- **Service alerts**: temporarily moved stops, unforeseen events affecting a station, route, or the entire network.

Updates of each type are provided in separate feeds, which are served via HTTP. The three types are stored in so-called Protocol Buffers (PB). Advanced routing engines can integrate both static and realtime GTFS data.

The data in Fig. 1 (left) shows part of an extracted GTFS-realtime '*VehiclePositions*' message. It provides information about a vehicle driving a specific trip on a particular date (first group of lines). The position is stored as geographic latitude and longitude (second group). In the third group, the *current_status* property describes whether the transit vehicle is driving towards the next stop ("IN_TRANSIT_TO") or is at a stop ("STOPPED_AT"). The timestamp shows when the position was recorded, expressed as Unix epoch time.

In a '*TripUpdate*' message (Fig. 1, right) each trip consists of a start block showing trip id and start date. This is followed by a sequence of trip updates for all stops along that trip with arrival and departure delay information. A negative value means early arrival or departure. In analysed OVapi datasets many of the delay slots for stops are left empty, or filled with questionable values, which will be addressed in more detail further below. The OTP is capable of considering these future delay estimations when calculating a trip. In the data example below, delay information is truncated after one stop.

```
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"
}
```

```
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"
}
...
```

Fig. 1: GTFS-realtime '*VehiclePositions*' message (left) and '*TripUpdate*' message (right)

3 Realtime Data for the Netherlands Test Case

Realtime live-feeds are continuously updated at a certain time interval. Both, the *TripUpdates* and the *VehiclePositions* messages need to be decoded to make them readable as objects in the Java programming environment. In our study we use downloaded *VehiclePositions* files over a period of 13 hours to derive ground truth positions of transit vehicles, and to compute ideal (fastest) routes retrospectively, based on these positions. Further, we use *TripUpdate* files that were downloaded for the same time period to simulate different levels of network delay knowledge the traveller has when planning a route. The compressed information in the protocol buffer file was accessed using Java classes (generated using Google's Protocol Buffers compiler and the agency's gtfs-realtime.proto file – accessible at http://gtfs.ovapi.nl) to decode the realtime feed, and was afterwards stored in a spatial database. We used a PostgreSQL database with a PostGIS extension for spatial queries. The data structures from the GTFS feed and the protocol buffer messages are used to create

tables that store the static schedule information with its associated geographic information, as well as vehicle positions and trip updates data for retrospective testing purposes.

VehiclePositions data messages were collected in 10-second intervals for May 28th 2014, from 7:00 am until 8:00 pm local Dutch time. An example of a compiled sequence of vehicle positions for a single vehicle, based on multiple downloaded *VehiclePositions* protocol buffer messages, is provided in Table 1. It shows how the vehicle progresses from one stop to another, as indicated by the stop sequence field. Assuming that the *VehiclePositions* data files provide accurate corrections to scheduled arrival and departure times, the modified timetable allows reconstructing the exact time profile of a trip after the fact. Such knowledge would allow one to retrospectively compute the ideal (e.g. fastest) route between a trip origin and destination.

Table 1: Extract of a single trip from *VehiclePositions* table

TRIP_ID	CURR_ STOP_SEQ	CURR_ STATUS	STOP ID	TIMESTMP	X	Y
8775680	15	IN_TRANSIT_TO	46529	2014-05-28 13:01:02-04	4.868515	52.302048
8775680	16	STOPPED_AT	0	2014-05-28 13:01:12-04	4.87308	52.302795
8775680	16	STOPPED_AT	0	2014-05-28 13:01:22-04	4.87308	52.302795
8775680	16	STOPPED_AT	0	2014-05-28 13:01:32-04	4.87308	52.302795
8775680	16	IN_TRANSIT_TO	42461	2014-05-28 13:01:42-04	4.87308	52.302795

Next, delay information collected from *TripUpdates* messages was used to create a modified timetable based on the original GTFS static timetable. These delay predictions, which are stored in a database table as well, can be compared to observed delays, derived from *VehiclePositions* protocol buffer messages. Thus, the accuracy of *TripUpdates* for upcoming stops along a trip can be assessed. *TripUpdates* messages were downloaded on May 28th 2014, from 7:00 am until 8:00 pm every 30 seconds. In addition, the static GTFS feed and the OpenStreetMap (OSM) files for the Netherlands were downloaded for the same day as well.

Table 2: Summary statistics of collected data

GTFS feed		**GTFS-realtime feeds**	
routes	2383	trips in *TripUpdate* feed from '2014-05-28 03:00:00-04'	10451
trips	713,199	number of trips in *TripUpdates*	73947
shapes	4,665,512	*TripUpdate* entries	258,916,245
shapelines	9236	*VehiclePositions*	16,963,413
stop times	15,962,597	number of trips in *VehiclePositions*	72612
stops	72256	trips with same trip_id in *VehiclePositions* and TripUpdates	70144
Creation of modified stop_times			
trips without gaps			72010
trips complete from first to last stop_sequence			52296

Table 2 summarizes the amount of downloaded data for the test period. The *TripUpdate* feed for a given time stamp is available for 10451 trips in this example. Vehicle positions were available for 72612 trips, which is only about 10% of all trips (713,199) within that time period, revealing data scarcity. During the process of computing modified stop times using *VehiclePositions* data, a total of 52296 complete trips were written into the new modified timetable. This means that information for 7.3% of trips available in the static timetable was modified in that process. For the rest of the trips, trips in *VehiclePositions* files were either incomplete or did not contain coordinate information.

4 Quality Assessment of Realtime Data

4.1 *VehiclePositions* Data

The aim of collecting *VehiclePositions* data is to create a modified reference timetable with observed arrival and departure times reflecting ground truth. The more vehicles publish their positions, and the more accurate these published positions are, the more accurate and complete the modified timetable is. If a station along a trip did not contain "STOPPED_AT" information, this could be because the vehicle did not stop at a station. In this case arrival and departure times for that stop were interpolated to the nearest recorded vehicle position of that specific trip, and considered valid data for building the reference dataset. In other cases, a station was not mentioned along a trip at all (not even through "IN_TRANSIT_TO"), or the sequence of stops was not in ascending order. Such a situation was then considered a gap, and the corresponding trip was excluded from further computations.

For the visualization of travel delay patterns, delays were computed as the difference between original (static) timestamps and timestamps in the reference timetable. Based on this, average delays were computed for each stop, where such delay information could be derived. This was the case for 41218 stops, out of 72256 total stops in the network. This means that about 43.0% of stations did not have a single delay value from any of the trips passing through, showing that OVapi real time *VehiclePositions* data in its current stage of implementation is far from complete.

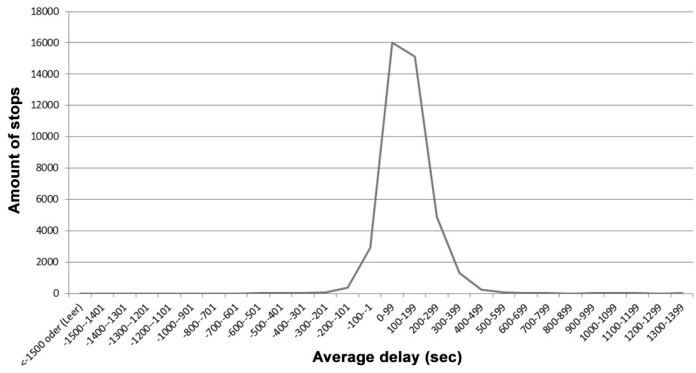

Fig. 2: Distribution of average delay at all affected stops on 05/28/2014

Out of all available average stop delays, 16004 stops (38.8%) had an average delay between 0 and 99 seconds, followed by 15114 stops (36.7%) with an average delay between 100 and 199 seconds (Fig. 2). Average delays were highest between 9am and 10am in the morning, with another peak in the evening rush hour between 5pm and 6pm.

The map in Fig. 3 is based on average available delay information for individual stops (shown as beige circles). Average delays were highest in the central parts (around Utrecht) and the northern parts (around Leeuwarden) of the Netherlands.

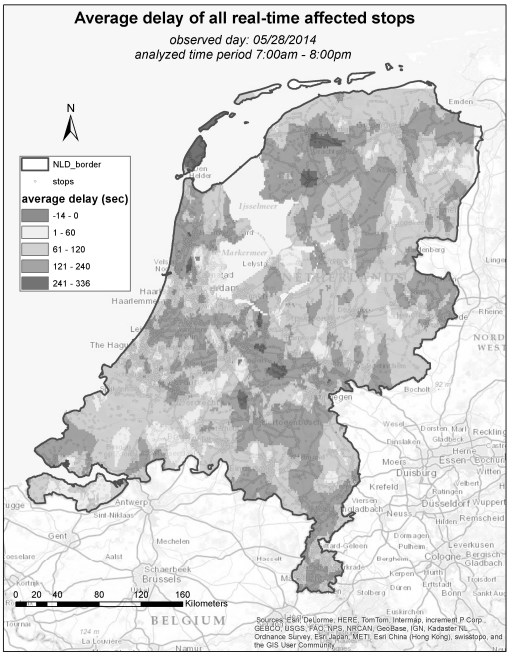

Fig. 3: Average delays derived from *VehiclePositions* feeds

4.2 *TripUpdates* data

TripUpdates contain estimated delay information for future stops as well as for past stops. For only 10.4% of all trips, *TripUpdate* feeds were available, and if so, they were incomplete. As an example, Fig. 4 shows the delay information for all stops on a single trip, extracted from three *TripUpdate* files at different times, and a single *VehiclePositions* file. The trip runs from 13:37 to 14:34. The information of the green line was extracted from the *TripUpdate* file at 13:40:31, i.e. after the trip started. This is when this trip was mentioned for the first time in a *TripUpdate* file. The purple line refers to delays in the TripUpdate feed downloaded at 14:00:01, and the blue line to a feed download at 15:00:01, after the end of the trip. The red line refers to quasi-true delay information extracted from *VehiclePositions* data. This example demonstrates poor data quality of the TripUpdate feed. That is, the *TripUpdate* information is not available until the trip begins, and after that, delay predictions are only available for about 10 minutes in the future, and set to zero after this.

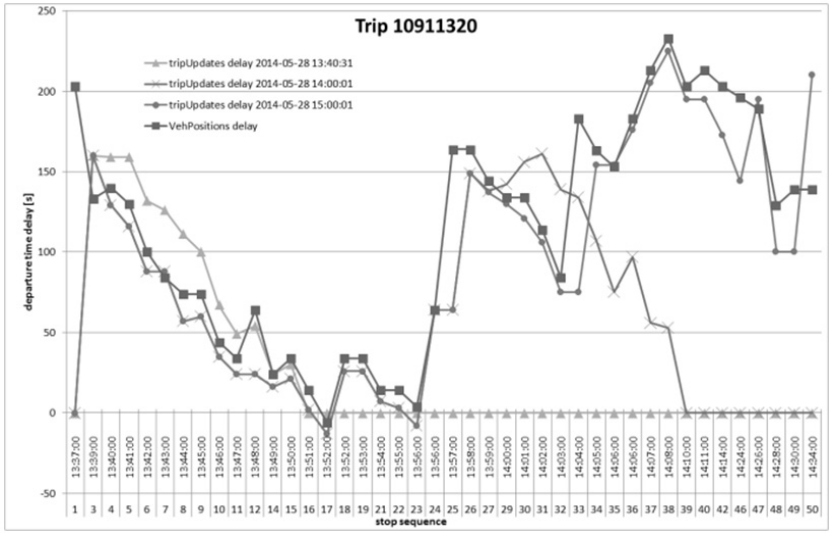

Fig. 4: Observed delays on May, 28th 2014 for *VehiclePositions* and delay predictions from *TripUpdates*

5 Route Analysis

5.1 Routing Software and Scenarios

The OTP is an open-source multi-modal routing planner that considers various transportation modes, such as bicycle, train, bus, or on foot. Its functionality can be customized with advanced routing options, e.g. by combining indoor with outdoor routing (WEYRER et al. 2014). For this research, the OTP was used to compare modelled routing results regarding availability of different levels of information to the traveller at the beginning of a trip and during a trip. These different levels of information are described in five conceptual scenarios. However, not all of the scenarios could be implemented due to current OTP limitations. Additionally, any full and comprehensive scenario performance comparison would require a complete set of *VehiclePositions* files, which are currently not covered by OVapi data. Therefore, we are limiting this part to selected representative routes as an initial proof of concept.

Scenario 1: Print trip on paper, not considering *TripUpdates*

In the first scenario, a user calculates a trip using the OTP without any delay information from *TripUpdates* files. It is assumed that the user prints the route and attempts to exactly follow the written instructions. If the traveller transfers to the next bus line and misses the planned bus, he or she has to wait for the following bus on that bus line. For this use case, the algorithm compares the static arrival and departure times from each bus on the calculated route to the observed (true) times that are stored in the reference timetable.

Fig. 5 shows a subset of the steps involved in the route computation for 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. In the second phase (num_calculation = 2, from line 15 on), the algorithm again begins at the start point, and compares the static start_time (i.e. departure time) and end_time (i.e. arrival time) with observed times from the reference timetable for all segments along the trip. Up to row 25, no bus would have been missed under the original schedule. However, since trip 11373739 arrived at 11:59:47 (see end_time column, highlighted) instead of the planned time of 11:59:00, which is also when the connecting bus departs (not shown in the table), the passenger would have missed this connecting bus. Therefore, the next scheduled bus (11379754) that departed at 12:00:00 was chosen instead of the planned bus (11372865). Bus 11379754 arrived at 12:09:00 and the initially computed walking time was added to this (line 26). Despite the delayed transfer, the trip itself, if followed in this manner, would have been about 2 minutes faster than the initial calculation, due to the earlier than planned arrival of the bus on the last segment (10389450).

	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
...							
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
...							
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

route based on static GTFS information

route based on observed vehicle positions

Fig. 5: Route example from scenario 1

Scenario 2: Print trip on paper, considering *TripUpdates*: The same scenario as before, except that the initial calculation considers *TripUpdates* available at the time of planning.

Scenario 3: Smartphone without realtime route computation: The passenger has a smartphone, which is used for the initial route calculation. If the traveller misses a connecting bus the smartphone is used to re-calculate a route from the current position.

Scenario 4: Smartphone including realtime route computation: The traveller's smartphone has access to the *TripUpdate* feed, and calculates the fastest route when passing by a stop along the planned trip, using the most recently downloaded *TripUpdates* file. In the case of a new faster route, e.g. due to delays, the application suggests the new route to the user.

Scenario 5: Optimal route calculation using modified GTFS feed: Although not possible for a traveller during navigation, this scenario computes the fastest route that would have been computed by the passenger if he or she had all the true future delay information while planning the trip. This scenario uses the reference timetable based on *VehiclePositions* files. The modified stop_times (described in section 4.1) were exported into the GTFS feed, which was used to create the routing graph in the OpenTripPlanner. In this scenario, the

initial route calculation (num_calculation = 1) already includes the quasi-true arrival and departure times.

The route request is the same for all scenarios. However, the execution of the route search differs by the used routing graph, the use of *TripUpdates,* and the use of recursive resubmissions of the trip request along the trip. Scenarios 3 and 4 (simulating a smartphone) could not be executed within the current OTP framework because of the following issue: If, according to the corrected reference table, a bus departs later due to a delay, the smartphone is supposed to consider that delayed departure time in its new route request. However, the OTP routing engine cannot currently handle such dynamic delay from the reference table, since it only has access to the timetable from the static GTFS feed.

5.2 Route Example

Regarding the original idea to statistically evaluate the effect of the different achievable scenarios, we randomly selected 45 origin-destination pairs in the northern part of the study area. However, due to the discovered data scarcity, a comprehensive assessment was not possible. For example, each of the 45 computed routes for scenario 2 (which considers available *TripUpdates* at the beginning of the trip) included several different bus lines (trips), giving a total of 391 trips. Only 23 out of these 391 single trips have realtime delay predictions from *TripUpdates*. Out of these, only 4 trips actually had a future delay, while all others had a delay of zero for future stops. Thus all routes in scenario 1 and 2 were identical. As opposed to this, some differences could be found between scenario 1 and 5. Fig. 6 shows the routes obtained for scenarios 1 (light grey) and 5 (dark grey) for the same origin and destination. The optimal (fastest) route would have saved 24 minutes of traveling.

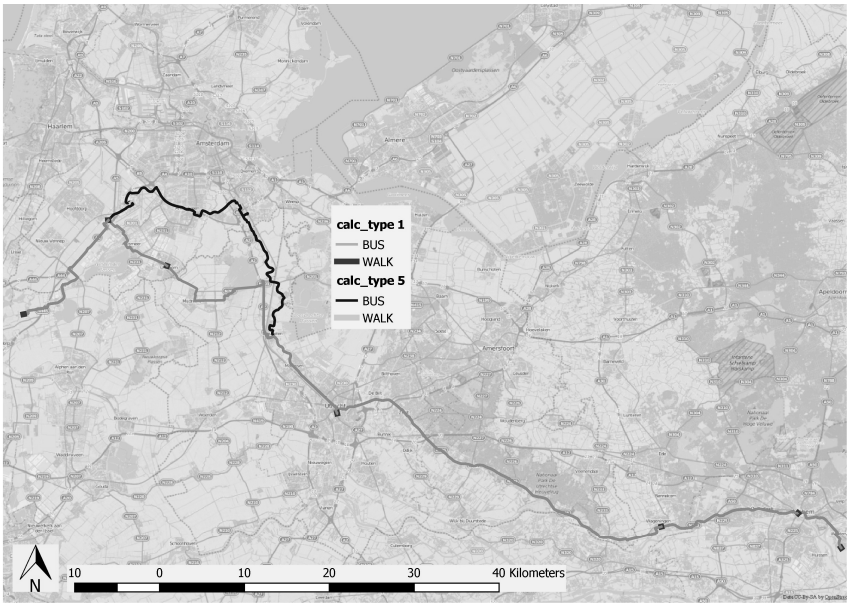

Fig. 6: Routes generated for scenario 1 and 5

6 Conclusions

This study analyzed the data quality of realtime public transit data provided as GTFS-realtime feeds for the Netherlands. Results show that at least up to May 2014 the data quality was not satisfactory, and that a large percentage of vehicle position and trip delay data was missing, which would be necessary for reliable realtime trip planning applications. One of the goals was to analyze user benefits of realtime information for trip planning, but, besides data scarcity, this was impossible due to OTP data handling. For example, the integration of *VehiclePositions* data into the OTP for scenario 5 caused problems by sometimes returning sub-optimal routes. Further, the assessment of usability for smartphone users with repeated computations along the trip (scenario 3 and 4) would require a modification of the OTP code. In the near future, we expect to see more transit agencies sharing realtime information, and their data should also be evaluated in order to assure positive effects on reliable multimodal route planning results.

Acknowledgements

This research was supported in part by a research grant awarded to the author Daniel Steiner by the Austrian Marshall Plan Foundation.

References

DESSOUKY, M., HALL, R., ZHANG, L. & SINGH, A. (2003), Real-time control of buses for schedule coordination at a terminal. Transportation Research Part A: Policy and Practice, 37 (2), 145-164.

EL-GENEIDY, A. M., HORNING, J. & KRIZEK, K. J. (2010), Analyzing transit service reliability using detailed data from automatic vehicular locator systems. Journal of Advanced Transportation, 45 (1), 66-79.

JARIYASUNANT, J., CARREL, A., EKAMBARAM, V., GAKER, D., KOTE, T., SENGUPTA, R. & WALKER, J. L. (2012), The Quantified Traveler: Using personal travel data to promote sustainable transport behavior. Proceedings of Transportation Research Board – 91st Annual Meeting, Washington, D.C., Jan 22-26, 2012, Transportation Research Board of the National Academies.

ROSSETTI, M. D. & TURITTO, T. (1998), Comparing static and dynamic threshold based control strategies. Transportation Research Part A: Policy and Practice, 32 (8), 607-620.

ROUSH, W. (2012), Google Transit: How (and Why) the Search Giant is Remapping Public Transportation. http://www.xconomy.com/san-francisco/2012/02/21/google-transit-a-search-giant-remaps-public-transportation/5/ (4/13/2015).

WEYRER, T. N., HOCHMAIR, H. H., & PAULUS, G. (2014), Intermodal Door-to-Door Routing for People with Physical Impairments in a Web-based Open Source Platform. Transportation Research Record: Journal of the Transportation Research Board, 2469, 108-119.