Dynamic Transfer Patterns for Fast Multi-modal Route Planning

Thomas Liebig, Sebastian Peter, Maciej Grzenda, Konstanty Junosza-Szaniawski

Abstract Route planning makes direct use of geographic data and provides beneficial recommendations to the public. In real-world the schedule of transit vehicles is dynamic and delays in the schedules occur. Incorporation of these dynamic schedule changes in multi-modal route computation is difficult and requires a lot of computational resources. Our approach extends the state-of-the-art for static transit schedules, Transfer Patterns, for the dynamic case. Therefore, we amend the patterns by additional edges that cover the dynamics. Our approach is implemented in the open-source routing framework OpenTripPlanner and compared to existing methods in the city of Warsaw. Our results are an order of magnitude faster then existing methods.

1 Introduction

In a changing world geo-spatial data is subject to dynamic changes and geo-information systems are required to incorporate real-time updates in their analysis and computations (Schnitzler et al. 2014). In this paper we focus particularly on route planning systems. While in a static world a bunch of algorithms exist to compute (shortest) paths from a starting location to a target location efficiently (compare Section 2), this problem becomes more difficult in case of multi-modal trip planning

Thomas Liebig (✉) · Sebastian Peter
TU Dortmund University, Dortmund, Germany
e-mail: thomas.liebig@tu-dortmund.de

Maciej Grzenda · Konstanty Junosza-Szaniawski
Warsaw University of Technology, Faculty of Mathematics and Information Science, Warsaw, Poland
e-mail: M.Grzenda@mini.pw.edu.pl

Konstanty Junosza-Szaniawski
e-mail: K.Szaniawski@mini.pw.edu.pl

including public transport, as temporal constraints, e.g. transit times and departure times, need to be incorporated. In real world, these static schedules are not met, but delays occur (Mazimpaka and Timpf 2016) and deviations from the schedule could be observed. Incorporation of these dynamic information in route computation is beneficial, as it allows to provide tractable travel recommendations to the public. The dynamic information on the delays can be achieved by monitoring positions of the vehicles and even by prediction of future delays. This enables pro-active trip computation.

In (Liebig et al. 2017, 2014), we highlighted how vehicular traffic predictions can be incorporated into trip computations. The paper at-hand incorporates public transport delays (which could be the result of prediction) and focuses on the tractability of dynamic transit computation. Existing single source shortest path computation algorithms for the dynamic transit problem suffer from their long computation time, the very fast route planning algorithm for transit networks, Transfer Pattern, does not guarantee soundness in case of real-time delay information. Our approach, overcomes these shortcomings and introduces dynamic transfer patterns, a data structure that encodes which novel transit possibilities are enabled due to the delays.

In a comparison with existing dynamic transit routing schemes, in the city of Warsaw, we highlight the performance gain using our method. Our findings are implemented in the commonly used open source trip planning framework OpenTripPlanner and a pre-configured Virtual Machine is ready to use in industrial context.

The paper is structured as follows. Section 2 provides an introduction to routing algorithms and the transit routing problem. Section 3 presents our Dynamic Transfer Pattern method. Implementation details are provided afterwards in Section 4. Next, we analyse tram delays in the city of Warsaw, Poland, and we continue with performance evaluation in this city, Section 4. In the end, we discuss future research ideas, Section 7.

2 Related Work

In this paper we focus on the point-to-point shortest path problem (Bast et al. 2016), where in a graph $G = (V,E)$ a path between a source $s \in V$ and target $t \in V$ needs to be found such that the cumulative edge-wise cost $l(u,v), with(u,v) \in E \subseteq V \times V$ along the path is minimized.

2.1 Shortest Path Routing

Standard solution to the problem is using Dijkstra's algorithm (Dijkstra 1959). Given the graph $G = (V,E)$ and $s,t \in V$, it initializes a queue of nodes $Q = V$ and a distance function over $V \times V$ with $dist(s,s) = 0$ and $dist(s,v) = \infty, \forall v \neq s, v \in V$. Until the queue is empty the node u with the smallest distance $dist(s,u)$ is picked

and removed from Q. For each neighboring node of u the distance is updated as follows: $dist(s,v) := dist(s,u) + l(u,v)$, if the latter is smaller than the former. Dijkstra's algorithm can be sped up by running it simultaneously from both s and t until a common node u is hit. In the slightly modified version of Dijkstra's algorithm A* (Hart, Nilsson and Raphael 1968) the order in the priority-queue for the traversal not only depends on the cumulated costs to reach a vertex in the graph but also on the expected costs to reach the goal from this vertex. Bound by Minkowski's inequality, whereas $||x+y||_p \leq ||x||_p + ||y||_p$ (known as triangle inequality for $p = 2$), A* prunes the search space in comparison to Dijkstra's Algorithm. A sound heuristic for the remaining cost estimation is the geographical distance that is always lower than the road-based distance.

In case of static cost functions Geisberger et al. propose a data structure called contraction hierarchies (Geisberger et al. 2008), which speed up the A* algorithm and enable trip calculation in large traffic networks at European scale. Instead of searching the shortest path directly within the traffic network, contraction hierarchies reduce the search space to the most important ones. In a preprocessing step these important segments are identified (based on the topology) and the network is extended by edges between these important links.

2.2 Shortest Path Routing in Transit Networks

In contrast to regular road networks, public transportation data enhances a spatial graph with temporal data by adding timetable information. A trip T serves a sequence of stops $stops(T) = (s_1, \ldots, s_n), s_i \in S$. Thus T connects two stops s_a and s_b if and only if $stop(T, s_a) < stop(T, s_b)$. If one or more trips contain the exact same sequence of stops, they form a line (Bast, Sternisko and Storandt 2013).

Common approach is to incorporate dynamic information into the graph G and then to apply Dijkstra's algorithm. This results in a time extended or time dependent model. In the time extended model every transit node is split into multiple vertices for each event (arrival, transit and departure). The time dependent model assigns every transit node one vertex and arcs encode temporal constraints.

A recent data structure and algorithm, *transfer patterns*, introduced by (Bast et al. 2010) is considered state-of-the-art in public transport routing. Based on the assumption that during a day, there are only a few optimal routes from stop s_s to stop s_t that differ only in the time they take place. In a preprocessing phase, optimal routes are computed as a sequence of transfer stations, neglecting the time component as well as information about intermediate stations. For each origin and target destination a directed acyclic graph is saved, containing all routes starting with the destination and containing all intermediate stations until the origin is reached.

In a realistic route planning scenario, various delays occur amongst the public transport vehicles. In contrast to vehicular traffic, trams and trains can not overtake, and vehicles in transit networks wait for each others (e.g. connecting trains), this causes delays to propagate differently than vehicular traffic jams. In addition, two

modes of transportation may share the same physical resource (e.g. buses or trams riding on vehicular street). Thus, two forms of delays in transit networks are distinguished in literature: 1) a vehicle is late due to own reasons, and 2) other vehicles are late caused by the former (Müller-Hannemann and Schnee 2009).

Several models for transit delays are reported in literature. The work in (Dibbelt et al. 2013) assumes independence. In contrast, (Goerigk et al. 2011) allow delays within their approach to cumulate. Sophisticated models incorporate dependencies among the vehicles into the delay (Higgins and Kozan 1998). In (Mazimpaka and Timpf 2016) the delays are analyzed visually.

In a trip planning application real-time predictions of delay are a main benefit as future delays may influence the route choice. Thus, we highlight two recent works on delay prediction and delay recognition: (Gal et al. 2015) applies queueing theory and assumes delays to aggregate, (Zygouras et al. 2015) detects delays and unexpected vehicle movement in real-time from the GPS traces.

In this work we do not focus on the prediction, but assume that we have information on delays of vehicles (in the commonly used GTFS realtime data format) either from vehicle observations or even predictions.

With such dynamics the trip computation becomes more difficult. Though states a previous paper (Bast, Sternisko and Storandt 2013) that transfer pattern are delay robust, but this only holds as long as no new transfers are enabled by the delay. In the likely case that novel transfers are enabled the existing transfer patterns do not represent this information and can not result in the optimal transit route.

3 Dynamic Transfer Pattern

Transfer Patterns were introduced in (Bast et al. 2010). The method comprises a data structure and an algorithm for fast transit route computation. In a preprocessing step all possible routes are precomputed and stored in a compressed way. For each public transport line a table is stored denoting in the columns the stops along the line. In this way it holds the maximal possible route without changes. The rows of the table represent the actual trips of the line, Table 1 gives an example, compare also (Cárdenas 2013).

Table 1 Transfer Pattern Example

line L17	$\vert s_a$	$\vert s_s$	$\vert s_b$	$\vert s_y$	$\vert \cdots$
trip 1	8:15	8:22 8:23	8:27 8:29	8:38 8:39	$\cdots$
trip 2	9:14	9:21 9:22	9:28 9:28	9:37 9:38	$\cdots$

In addition, for every station a list is stored with the passing lines and their position in the trips, see an example in Table 2.

Table 2 Transfer Pattern Example (continued)

s_s:	(L8,4)	(L17,2)	(L34,5)	(L87,17)	⋯
s_y:	(L9,4)	(L13,5)	(L17,4)	(L55,16)	⋯

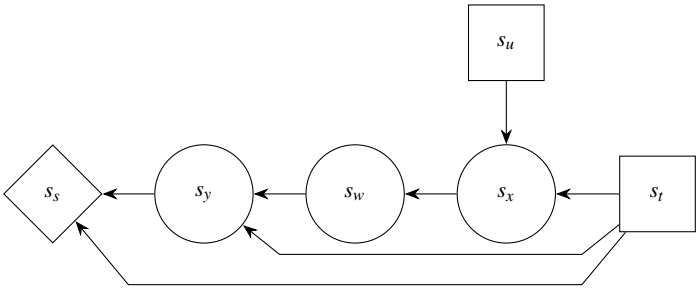

Fig. 1 DAG structure of the transfer pattern.

With Transfer Pattern a route from s_{start} to s_{stop} at time t is calculated by the intersection of the lists for s_{start} and s_{stop}, the first connection after time t is the desired result. As an example, the route from s_s to s_y at 9:03 is computed by intersecting the lines in Table 2, we find that line 17 connects the stops on positions 2 and 4. According to Table 1 the earliest possible trip is departing 9:22 at s_s and arrives at s_y at 9:37. In the pre-processing phase the shortest paths amongst all stops (neglecting temporal information) are constructed and intermediate stops are stored in a directed acyclic graph (DAG). Figure 1 exemplifies this for s_s.

In case of a routing query from stop s_s to stop s_t the DAG for s_s is picked and all connecting paths among s_s to s_t are taken into account for routing.

With dynamic schedule information at hand, the time table information (Table 1 in our example) can be updated and the route computation could be performed as in the regular case (Bast, Sternisko and Storandt 2013). However, it might be that the optimal connection is not precomputed as new connections could be enabled by the delay itself, especially if multiple trips arrive belated. Incorporation of the delays in trip computation provides benefits to travelers, thus next section addresses our implementation of dynamic transfer pattern in the routing platform OpenTripPlanner.

Our approach is to include alternative patterns that emerge due to delays or cancellations in the data structure by simulating these very delays during precomputation. In order to achieve this, all lines of non-final trips of a Transfer Pattern originating in s_s are recorded during computation of static routes. In a next step, these lines are combined into delay scenarios.

As a simple example of such a scenario, consider a route from s_s to s_t by transferring at s_y. Furthermore, the first trip is delayed considerably, making an alternative route with a transfer at s_v favorable in the Pareto sense. Since the advantage of the second pattern depends on a certain scenario of realtime delays, it could not have been precomputed as a static Transfer Pattern.

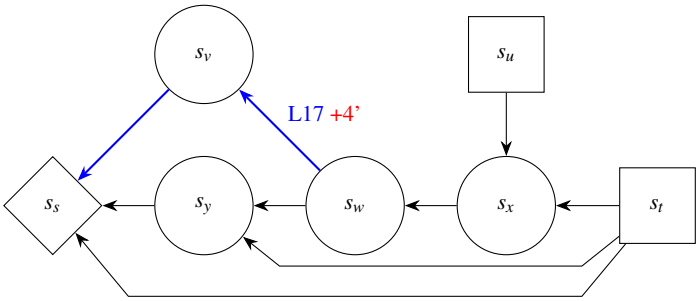

Fig. 2 DAG structure of the dynamic transfer pattern.

Alternative Transfer Patterns are computed by applying to the graph by delaying according trips artificially, making transfers to the next respective trips in time infeasible. When running the routing algorithm now, it returns the next best routes considering the delay situation. All such alternative Transfer Patterns are eventually merged into the regular DAGs after having been classified in terms of lines that were artificially delayed and their respective amount of delay. See Figure 2.

All lines used in regular Transfer Patterns can be combined into delay scenarios in numerous ways. Considering all combinations of n delayed lines producing 2^n scenarios implies enormous computational costs. Multiple ways of combining delayed trips were thus introduced, trying to cover many alternative routes while keeping the overall number of scenarios as low as possible.

A trivial approach is to incorporate delays of only a single line at a time. This method may already significantly increase computation costs for large graphs compared to merely computing static patterns. For this reason, picking a limited number of random lines per Transfer Pattern subgraph was introduced as another approach. Lastly, in pursuance of computing the most useful alternative routes, past data of lines with a high likelihood of delay can be utilized. This means picking often-delayed lines or combinations thereof more frequently when constructing a limited amount of delay scenarios.

When answering a routing query with dynamic traffic data, the corresponding query graph is fetched in a similar fashion regular Transfer Patterns are handled. When walking across the graph from target to source, the delay classification of each arc is checked. Arcs with no delay classification are always considered, in contrast to arcs with delay classification, which have to match the actual traffic situation. Realtime traffic information match a classification if and only if each trip of the classification is delayed by an amount bigger or equal to the amount specified.

Since real time delay information has been applied to departure and arrival times, it is possible that some patterns need a much longer travel time or are completely infeasible and thus no longer interesting to the user. These patterns are discarded either when direct connections are fetched for all trips or when they are dominated by other patterns in the Pareto sense.

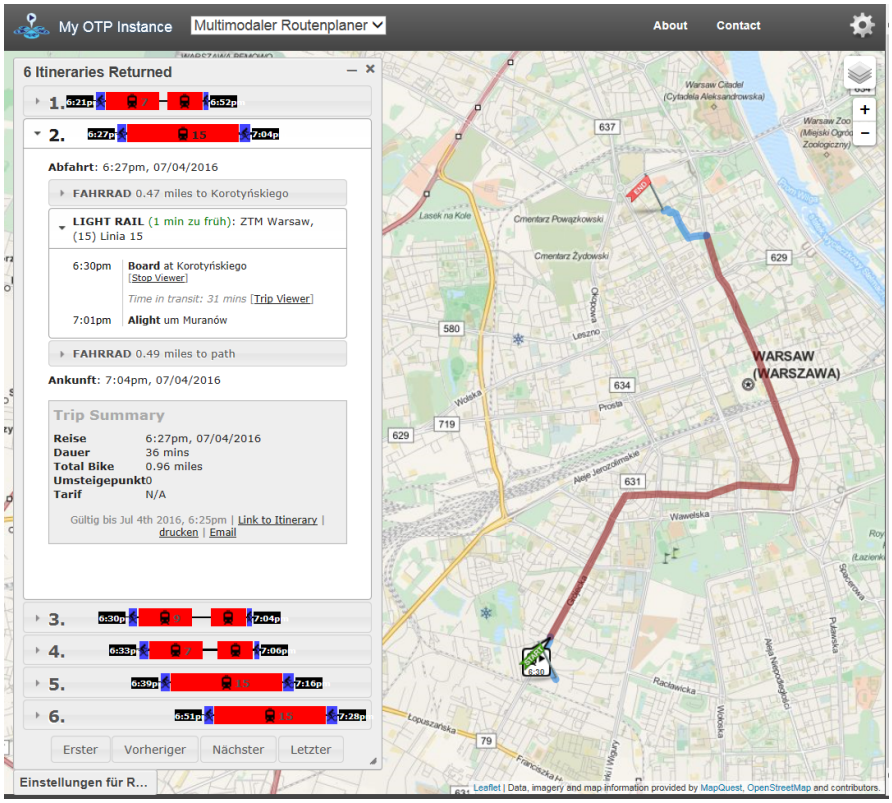

Fig. 3 OpenTripPlanner incorporating dynamic delay information.

4 Integration in OpenTripPlanner

We implemented the hereby presented dynamic transfer pattern routing scheme in OpenTripPlanner (a commonly used open-source trip computation framework). Therefore we consume information on the transit network and schedules from a commonly used GTFS representation, and dynamic updates in GTFS-realtime format The latter could be retrieved either by an automatic vehicle location system, as in (Mazimpaka and Timpf 2016) or by real-time predictions as in (Zygouras et al. 2015). OpenTripPlanner uses the street network provided by OpenStreetMap. Our routing scheme is integrated as an optional routing algorithm in OTP, its sources are publicly available and a running setup is preconfigured as vagrant box[1]. A screenshot of the user-interface of the routing system in depicted in Figure 3.

[1] https://bitbucket.org/tliebig/developvm/branch/transferpatterns

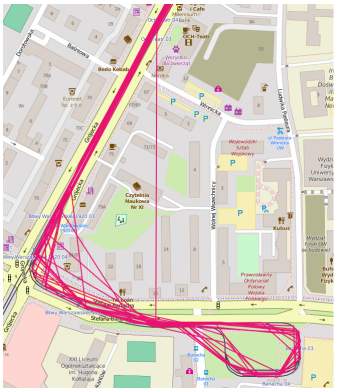

Fig. 4 Sample tram routes approximated from GPS data. Periodical location reporting and gaps in location reporting result in occasionally significant distance between consecutive tram coordinates. Best viewed in color.

5 Analysis of Tram Location Data

It is vital before delays are incorporated in route planning to understand some delay data. In our case, we study the delays in the city of Warsaw, Poland. These data are attained through the integration of location data retrieved in near real time manner from GPS sensors present in trams, tram stop point coordinates and schedule data. Every tram reports its GPS position together with its identifier, which is a combination of line number and brigade number. Such reports are produced every 30 sec. These data are available via API to the public, except for the number of brigade, which was made available by the City of Warsaw for the project. The data are not clean, various types of problems appear: some records are missing, GPS positions are inaccurate, occasionally two trams report the same identifier. What should be emphasised here is that no imputation of missing data was applied. As an illustration of GPS data quality issues, see the approximation of the tram routes provided in Figure 4.

Even if the GPS data were perfect, it would not be clear how to extract from GPS data precise times of arrival and departures of trams at tram stops. One of the reasons is that a tram stop is defined by point coordinates of the stop. The simplest method to compute the departure of a tram from a stop is to compute the time when a tram leaves a circle centered at stop point. However, it is not clear what radius of the circle is appropriate. If it is too small, then a tram can 'miss' the stop. The radius must be larger than the length of a tram, since two trams in a row can stop at the same time on the same tram stop. On the other hand, if the radius is big then other problems appear. The tram might have left the stop but is waiting near the stop by the traffic lights, and we still consider it as present at the tram stop. In quite many cases, a tram stop is located immediately before traffic lights. In such cases, the fact that a tram does not leave the stop on time may mean delay caused by the tram

itself or may be due to traffic lights suspending tram departure. All these factors contribute to the fact that precise calculation of arrival and departure times would not be possible even based on accurate tram coordinates i.e. coordinates not affected by GPS inaccuracies.

To address these problems, at least partially, for every tram stop we define a line going through the stop point. If a tram crosses this line, we consider the tram has left the tram stop. If the route of the tram near the tram stop is straight, the line is perpendicular to the route. If the tram stop is near the turn of the route, then the line is parallel to the bisector of the turn. These rules let us address the issue of estimating tram departure time based on location time series. Still, departure time estimates remain noisy for the reasons described above. Moreover, they can be affected by traffic lights. For these reasons it is inevitable for the trams to:

- arrive earlier than needed, because of varied traffic light conditions
- be considered late because of standing behind another tram (and tram stop line), while being already at tram stop and having its doors open
- be considered late because of waiting for traffic light change.

Finally, what is worth mentioning here is that temporary conditions such as events causing major traffic disruption may be not reflected in the schedules and are just announced through City of Warsaw Twitter and RSS channels. Taking into account all these aspects, what is considered as a delay or early arrival may mean on time departure or departure at a time not matching the schedules due to the circumstances beyond control of Warsaw Trams.

To illustrate the distribution of the differences between scheduled time and the most approximate observed departure time in time domain, the data for 23rd of March 2016 coming from Warsaw City tram system was used. In the preprocessing stage, it was limited to the time between 4:00-23:59. Moreover, differences between planned and observed departure exceeding 20 minutes were isolated for further investigation. Altogether events related to these two categories correspond to less than 4% of the data. Based on the remaining 96.4% of the data, the analysis described below was performed. First of all, Fig. 5 illustrating all differences has been developed. Not surprisingly, it is dominated by on-time arrivals. It is more interesting to look at early departures and delayed departures, provided in the left and right part of Fig. 5, respectively. In particular, what can be observed is quite a significant proportion of early departures. As stated above, based on the available data distinction between actual arrival and departure time may be problematic. Hence, the interpretation of the histograms necessitates particular attention being paid to the way the data has been collected and processed.

What is of particular interest for dynamic route planning is whether delays and early departures vary over the day. Fig. 6 answers this question showing mean absolute delay for individual hours of the day. Quite surprisingly, lower values are observed for peak hours 7-9 and 15-18. This may suggest the potential for both the development of prediction module and schedule improvement. Further analysis performed for individual tram lines separately is provided in Fig. 7. This reveals that

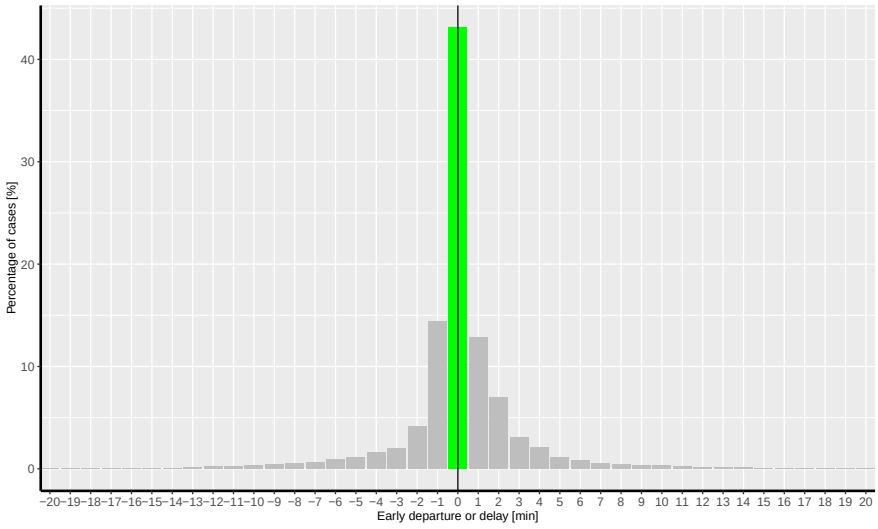

Fig. 5 Percentage of share of differences between planned and observed time of passing tram stop line; the bar in green shows the percentage of cases of less than 30 seconds difference. Based on the data for 23rd of March 2016 from Warsaw City trams, limited to the time between 4:00-23:59 and containing coordinate records from all 25 tram lines operated in this period. It can be observed that majority of trams pass tram stop lines in time. Still, relatively many early and delayed departures are observed. Among other reasons, the aforementioned limited ability to precisely determine arrival and departure time contributes to the problem. The figure is best viewed in color.

two tram lines largely contribute to the overall mean absolute delays. Hence, some routes are far more susceptible to delays than others.

Another performance indicator to consider is the proportion of trams on time, departing early and late. We consider a tram to be on time, if it passes tram stop point at most 120 seconds before or 120 seconds after the scheduled time. The percentage of early, on time and late departures of trams throughout the day remains largely stable. It is surprising that the trams are relatively more punctual in the rush hours and less punctual just before the morning peak and just after the afternoon peak. In particular, minimum percentage of on time departures per an hour is 75.6% and occurs at h=21. On the other hand, maximum percentage of on time departures per an hour is 85.2%, which is observed at $h = 17$ i.e. during peak afternoon period. The reasons of this are worth investigating in the future.

The preliminary analysis of tram location data compared with schedule data, reveals that:

- departure time not matching schedule time can be identified, but has to be analysed carefully, taking into account limited certainty of departure time estimation,
- still, noticeable number of early and late departure events can be observed in the data,

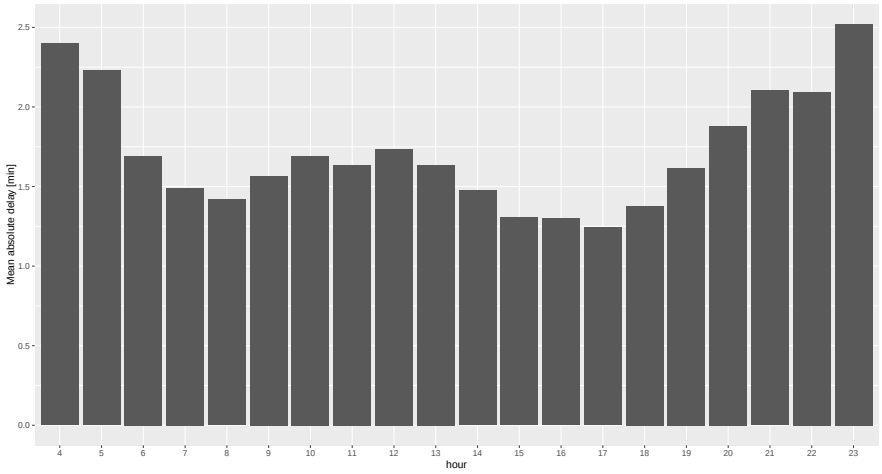

Fig. 6 Average mean absolute tram delay per an hour. Both early and late departures are considered. Significant variation of mean absolute difference between planned and observed departure takes place over the day. The differences observed are between approx. 1.3 and 2.5 min. Interestingly, the smallest differences meaning the most punctual trams are observed during peak traffic hours i.e. for $h \in \{7,8,9,15,16,17,18\}$

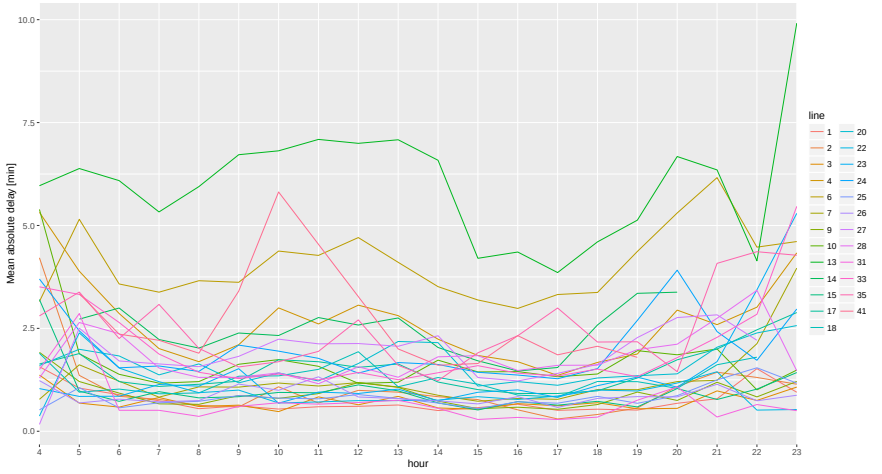

Fig. 7 Average mean absolute tram delay per an hour for individual tram lines. Both early and late departures are considered. Differences in the punctuality of individual tram lines are observed. These vary between 0.5 min and 10 mins depending on the line and time of the day. Best viewed in color.

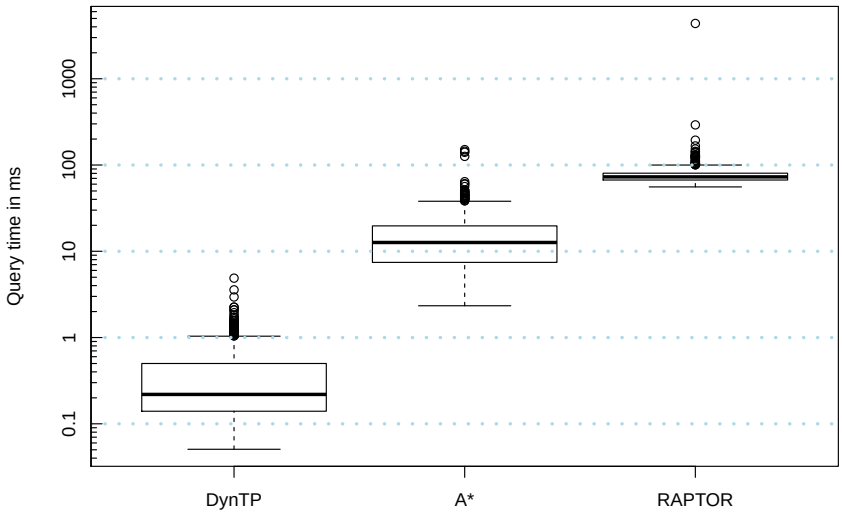

Fig. 8 Distribution of the computation time of public transport routing schemes (Dynamic Transfer Patterns, A*, RAPTOR) in milliseconds for 1000 randomly chosen source target locations in OpenTripPlanner under realistic conditions in Warsaw, Poland. Note logarithmic scaling.

- tram delays and early departures significantly vary based on the time of the day and tram line.

6 Performance Evaluation

Main goal of the paper hereby is to speed-up trip computation in case of dynamic delays. As we aim to apply the transit route computations in an industrial project, we decided to extend capabilities of existing open source platform OpenTripPlanner (OTP). Thus, we compare our dynamic transfer pattern with the transit routing schemes already available in OTP. In OTP the algorithms A* (Hart, Nilsson and Raphael 1968) and RAPTOR (Delling, Pajor and Werneck 2012) are available. We test the routing performance in the city of Warsaw, Poland. On startup, we perform initial pre-processing of the transfer patterns based on a GTFS timetable information. Afterwards, we compute for approximately 1000 source destination pairs the public transport routes. Experiments are performed on a regular desktop machine, computation time is measured in milliseconds. The resulting distribution of computation times can be seen in Figure 8, please note logarithmic scaling.

As can be seen in the Figure 8 our algorithm is an order of magnitude faster than existing transit computation schemes. However, initial pre-processing is exhaustive and required about 8 hours, this can easily be reduced by distribution of the initial preparations.

7 Discussion and Future Work

In this work, we focused on fast transit route computation in a changing world. We highlighted the shortcomings of existing algorithms, that are either very slow, or do not incorporate real-time transit information. We overcame these limitations by introduction of Dynamic Transfer Patterns. In this routing scheme, we applied the basic idea of (Bast et al. 2010) but created additional links in the patterns for transit connections that occur due to the delays. Thus, the modified Transfer Patterns can be applied also in dynamically changing environment. The method was made publicly available as ready-to-use virtual machine and as source code integrated in the commonly used OpenTripPlanner. As input our implementation depends on the commonly used GTFS and GTFS Realtime data structures that encode transit information.

The performance of our implementation in comparison to existing methods was measured using real-world data, our method achieved computation times that are an order of magnitude faster than existing ones. However, precomputation is quite exhaustive. For this step we propose future research on biasing the precomputations to the most prominent ones. Visual inspection of the delays, as performed in (Mazimpaka and Timpf 2016), can help to prioritize certain delay computations. Moreover the precomputation runs for every station separately, this step could probably benefit from parallelization. These two points are subject for future research.

Acknowledgements The authors received funding from the European Union Horizon 2020 Programme (Horizon2020/2014-2020), under grant agreement number 688380 “VaVeL: Variety, Veracity, VaLue: Handling the Multiplicity of Urban Sensors”.

References

Bast, Hannah, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner and Renato F. Werneck. 2016. *Route Planning in Transportation Networks*. Cham: Springer International Publishing pp. 19–80.

Bast, Hannah, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev and Fabien Viger. 2010. Fast routing in very large public transportation networks using transfer patterns. In *European Symposium on Algorithms*. Springer pp. 290–301.

Bast, Hannah, Jonas Sternisko and Sabine Storandt. 2013. Delay-robustness of transfer patterns in public transportation route planning. In *ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013*. Vol. 33 Schloss Dagstuhl Leibniz-Zentrum fuer Informatik pp. 42–54.

Cárdenas, Cynthia Jiménez. 2013. Efficient Multi-modal Route Planning with Transfer Patterns. Master's thesis Freiburg University.

Delling, Daniel, Thomas Pajor and Renato Werneck. 2012. Round-Based Public Transit Routing. In *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*. Society for Industrial and Applied Mathematics.

Dibbelt, Julian, Thomas Pajor, Ben Strasser and Dorothea Wagner. 2013. Intriguingly simple and fast transit routing. In *International Symposium on Experimental Algorithms*. Springer pp. 43–54.

Dijkstra, Edsger W. 1959. "A note on two problems in connexion with graphs." *Numerische mathematik* 1(1):269–271.

Gal, Avigdor, Avishai Mandelbaum, François Schnitzler, Arik Senderovich and Matthias Weidlich. 2015. "Traveling time prediction in scheduled transportation with journey segments." *Information Systems* .

Geisberger, Robert, Peter Sanders, Dominik Schultes and Daniel Delling. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*. Springer pp. 319–333.

Goerigk, Marc, Martin Knoth, Matthias Müller-Hannemann, Marie Schmidt and Anita Schöbel. 2011. The price of robustness in timetable information. In *OASIcs-OpenAccess Series in Informatics*. Vol. 20 Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Hart, Peter E, Nils J Nilsson and Bertram Raphael. 1968. "A formal basis for the heuristic determination of minimum cost paths." *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Higgins, Andrew and Erhan Kozan. 1998. "Modeling train delays in urban networks." *Transportation Science* 32(4):346–357.

Liebig, Thomas, Nico Piatkowski, Christian Bockermann and Katharina Morik. 2014. Predictive Trip Planning - Smart Routing in Smart Cities. In *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014*. Vol. 1133 CEUR-WS.org pp. 331–338.

Liebig, Thomas, Nico Piatkowski, Christian Bockermann and Katharina Morik. 2017. "Dynamic Route Planning with Real-Time Traffic Predictions." *Information Systems* 64:258–265.

Mazimpaka, Jean Damascène and Sabine Timpf. 2016. A visual and computational analysis approach for exploring significant locations and time periods along a bus route. In *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM pp. 43–48.

Müller-Hannemann, Matthias and Mathias Schnee. 2009. Efficient timetable information in the presence of delays. In *Robust and Online Large-Scale Optimization*. Springer pp. 249–272.

Schnitzler, François, Alexander Artikis, Matthias Weidlich, Ioannis Boutsis, Thomas Liebig, Nico Piatkowski, Christian Bockermann, Katharina Morik, Vana Kalogeraki, Jakub Marecek, Avigdor Gal, Shie Mannor, Dermot Kinane and Dimitrios Gunopulos. 2014. Heterogeneous Stream Processing and Crowdsourcing for Traffic Monitoring: Highlights. In *Machine Learning and Knowledge Discovery in Databases*. Vol. 8726 of *Lecture Notes in Computer Science* Springer Berlin Heidelberg pp. 520–523.

Zygouras, Nikolaos, Nikos Zacheilas, Vana Kalogeraki, Dermot Kinane and Dimitrios Gunopulos. 2015. Insights on a Scalable and Dynamic Traffic Management System. In *EDBT*. pp. 653–664.