CASPT 2015

Modelling Urban Public Transportation Networks to Support Robust Routing

Kateřina Böhmová · Matúš Mihalák ·
Tobias Pröger · Peter Widmayer

Abstract Given an urban public transportation network and historic delay information in a form of past observations, our goal is to suggest reliable journeys. We describe a theoretical model for the network and discuss challenges that arise when modelling real-world situations.

Keywords Modelling · Robustness · Route planning · Public transportation

1 Introduction

Motivation. Consider a dense public transportation network in which buses, trams, metros, etc. operate with high frequency. Imagine that we would like to travel from a departure stop d to a target stop t, and that it is important to arrive at t no later than at time t_A. Determining the right moment to leave d is nontrivial: Although we want to arrive at t not later than at time t_A, we don't want to leave d much too early. In an ideal situation, every bus and every tram is on time, and it is sufficient to compute a journey that is planned to leave d as late as possible but still reaches t at the latest at t_A. However, an empirical study performed by Firmani et al (2013) on the transportation network of Rome indicates that the timetable information and real movement

This work has been partially supported by the Swiss National Science Foundation (SNF) under the grant number 200021 138117/1, and by the EU FP7/2007-2013 (DG CONNECT.H5-Smart Cities and Sustainability), under grant agreement no. 288094 (project eCOMPASS). Kateřina Böhmová is a recipient of the Google Europe Fellowship in Optimization Algorithms, and this research is supported in part by this Google Fellowship. Parts of this work appeared in Böhmová et al (2013).

K. Böhmová, T. Pröger, P. Widmayer
Department of Computer Science, ETH Zürich, Switzerland
E-mail: katerina.boehmova@inf.ethz.ch, tobias.proeger@inf.ethz.ch, widmayer@inf.ethz.ch

M. Mihalák
Department of Knowledge Engineering, Maastricht University, The Netherlands
E-mail: matus.mihalak@maastrichtuniversity.nl

of the vehicles (based on GPS data) are only mildly correlated. We performed a similar study on the public transportation network of Zürich and were able to observe a comparable behaviour. Thus, routing based solely on a scheduled timetable without considering the occurrence of delays may lead to solutions of quite an unsatisfactory quality. Since in reality we always should expect delays, we study the problem of finding *robust* journeys from d to t that likely arrive before time t_A, but still leave d at a "reasonable" time (i.e., not much earlier than necessary).

Why real-time information alone is not sufficient. Since nowadays many network operators provide real-time information that indicate the current position of the vehicles (and also current delays), one might be tempted to argue that the computation of *robust* journeys is not really essential any more. However, we do not share this point of view. First, real-time information does not help if the journey is planned some time in advance (such as, e.g., few hours earlier). Second, in reality it often happens that delays occur suddenly and can not be foreseen in advance, especially not at the time when the journey is planned. For example, consider an *dt*-journey that consists of two lines l_1 and l_2, and imagine that the transfer time between the lines is 2 minutes. Even if l_1 leaves d on time, every upcoming delay of more than 1 minute (which might always occur) leads to a late arrival at t. Thus, considering just real-time data might not be sufficient in many situations.

Related work: Finding fast journeys. The task of finding a fastest (but not necessarily robust) journey in the planned timetable of a public transportation network has been extensively considered in the literature. Common approaches model the network as a graph and compute a shortest path in this graph (see Müller-Hannemann et al (2007) for a survey). The most widely considered graph-based models are the *time-dependent* (Brodal and Jacob (2004); Nachtigall (1995); Orda and Rom (1990, 1991)) and *time-expanded* (Müller-Hannemann and Weihe (2001); Pallottino and Scutella (1998); Schulz et al (2002)) models. Various improvements have been developed, and experimental studies suggest that these can also be used in practice (see, e.g., Bauer et al (2011); Delling et al (2009); Pyrga et al (2008)). Recent approaches (e.g., Delling et al (2012); Dibbelt et al (2013)) avoid the construction of a graph and process the timetable directly. For example, Delling et al (2012) described a strategy which is centred around transportation lines (e.g., train or bus lines). When considering the arrival time and the number of stops as criteria, it can be used to find all pareto-optimal journeys. Bast et al (2010) observe that for two given stops, we can find and encode each sequence of intermediate transfer stations (i.e., stations where we change from one line to another) that can lead to an optimal route. The set of these sequences of transfers is called *transfer pattern*. These patterns can be precomputed, leading to very fast query times.

Related work: Finding robust journeys. The problem of computing *robust* journeys in public transportation networks has been studied before. Many authors (e.g., Boyan and Mitzenmacher (2001); Dibbelt et al (2013); Frank (1969); Nikolova et al (2006)) model congestion using stochastic methods, e.g. they assume that stochastic delays on edges or vehicles are known. Disser et al (2008) extended Dijkstra's algorithm for computing pareto-optimal multi-criteria journeys using a given fixed timetable. They defined the reliability of a journey as a function depending on the minimal time to change between two subsequent trains, and used it as an additional criterion. Müller-Hannemann and Schnee (2009) introduced the concept of a *dependency graph* to predict secondary delays caused by some current primary delays, which are given as input. They showed how to compute an optimal journey with respect to these predicted delays. Goerigk et al (2011) assumed that a set of delay *scenarios* is provided, and showed how to compute a journey that arrives on time in every scenario (i.e., a *strict robustness* approach). This approach requires a feasible solution for every realisation of delays for every event. This is quite conservative, as in reality not every combination of event delays appears. Furthermore, they introduced the concept of *light robustness*, which aims to compute a journey that maximises the number of scenarios in which the travel time of this journey is not more than a fixed time worse than the optimum.

Our contribution. Instead of modelling delays explicitly, our goal is to infer delay information implicitly from historic traffic data. Also, we are interested in the robustness of complete *journeys*, and not on computing concrete delay distributions for concrete edges in the public transportation network. This is reasonable from the user's perspective: If our journey consisted of two lines l_1 and l_2, and if the transfer time between these lines was 2 minutes, then the information that l_1 is delayed by 2 minutes is only helpful if also the delay of l_2 is known. If, for example, all lines in the network were delayed by 2 minutes, then any planned transfer between two lines was also feasible in reality. The only difference between the real and the planned journey is that the overall travel time increases by 2 minutes, which might be acceptable. However, if only some lines were delayed so that certain planned transfers become infeasible in reality and has to wait a long time (e.g., an hour or even the whole night) for some vehicle, then this is a serious issue from the user's perspective.

Since our goal is to infer robust journeys from past data, we need a solution concept that allows journeys to be comparable over different past days. Thus, classical solutions concepts such as time-expanded or time-dependent graphs are not suitable any more. In Böhmová et al (2013), we introduced a new approach for finding robust journeys that investigates how journeys performed in past instances. Similarly to more recent approaches we try to exploit the problem structure explicitly (e.g., by considering lines) instead of implicitly modelling properties into a graph. In the present paper, we shortly describe our solution concept and then discuss some consequences of such a modelling–various issues arising when modelling real-world situations.

2 Model

Stops and lines. Let $\mathcal{S}$ be the set of stops of the public transportation network. A *line* is an ordered sequence $\langle s_1, \dots, s_k \rangle$ of stops from $\mathcal{S}$, where s_i is served directly before s_{i+1}, and $\mathcal{L}$ denotes the set of all lines. We explicitly distinguish two lines that serve the same stops but have opposite directions; see Section 4 for details. For $\beta \in \mathbb{N}_0$, a sequence of lines $r = \langle l_1, \dots, l_{\beta+1} \rangle \in \mathcal{L}^{\beta+1}$ is called an *dt-route* if there exist $\beta + 2$ stops $s_0 := d, s_1, \dots, s_\beta, s_{\beta+1} := t$ such that for every $i \in \{1, \dots, \beta + 1\}$, both stops s_{i-1} and s_i are served by the line l_i, and s_{i-1} is served (not necessarily directly) before s_i. For $i \in \{1, \dots, \beta\}$, we say that a *transfer* between the lines l_i and l_{i+1} occurs at the *transfer stop* s_i. Notice that in general there might exist multiple transfer stops between two lines. We also note that a route might contain a line $l \in \mathcal{L}$ multiple times. For a destination stop $d \in \mathcal{S}$, a target stop $t \in \mathcal{S}$ and an integer $\beta \in \mathbb{N}_0$, let $\mathcal{R}^{\beta}_{dt}$ denote the set of all *dt*-routes with at most β transfers.

Trips and timetables. While the only information associated with a line itself are its consecutive stops, it usually is realised multiple times per day. Each of these concrete realisations is called a *trip*. A *timetable* stores for every stop $s \in \mathcal{S}$ all daily arrival and departure times of every trip that contains s. Since we want to learn from past observations, we assume that we have a set $\mathcal{T}$ of *recorded* timetables T_i that describe how various lines were operated during a given time period (e.g., on a concrete day).

In many networks, there also exists *planned* timetable $T_{planned}$. We assume it to be periodic, i.e., every line realised by some trip τ will be realised by a later trip τ' again (not necessarily on the same day). If such a timetable is available, then the recorded timetables in $\mathcal{T}$ are concrete executions of the planned timetable.

In the following, *timetable* refers both to the planned as well as to a recorded timetable. We assume that timetables respect the FIFO property, i.e. two buses or trams *of the same line* do not overtake each other.

Goal. Let $d, t \in \mathcal{S}$ be the departure and the target stop, $\beta \in \mathbb{N}_0$ be the maximum number of allowed transfers, $\mathcal{R}^{\beta}_{dt}$ the set of all *dt*-routes with at most β transfers, t_A the latest allowed arrival time at t, and $\mathcal{T}$ a set of recorded typical timetables for comparable time periods (e.g., daily recordings for the past Mondays). A *journey* j consists of a departure time t^j_D, a route $r_j = \langle l_1, \dots, l_{\alpha+1} \rangle \in \mathcal{R}^{\alpha}_{dt}$ for some $\alpha \leq \beta$, and a sequence of transfer stops $\langle s_1, \dots, s_\alpha \rangle$. The intuitive interpretation of such a journey is to be physically present at the departure stop d at time t^j_D, take the first arriving (vehicle of) line l_1, and for every $i \in \{1, \dots, \alpha\}$, leave line l_i at stop s_i and take the next arriving line l_{i+1} immediately. Our goal is to use the recorded timetables in $\mathcal{T}$ (and the planned timetable, if available) for computing a travel recommendation in form of one or more (robust) journeys from d to t that will likely arrive on time (i.e., at time t_A, or earlier) on a day for which the concrete travel times are not known yet.

For obtaining meaningful results (i.e., recommendations that really arrive on time) in real-life applications, it is important that the situation in which the timetables in $\mathcal{T}$ were obtained is comparable to the situation for which the recommendation is computed. This of course does *not* mean that the timetables in $\mathcal{T}$ should not contain too many delays (otherwise there wouldn't be a need for robust routing), but extern circumstances such as the day of the week or the season should be comparable. For example, road traffic on Wednesdays is different from the traffic on Mondays or Fridays, hence $\mathcal{T}$ should only contain timetables that were recorded on the same day of the week as the day for which a journey recommendation is requested. Also, traffic in winter usually differs substantially from traffic in summer due to different weather conditions. In this paper, we will not pursue the issue of identifying typical situations further.

3 Computing Robust Journeys

Overview. For computing computing robust dt-journeys, the first step consists in listing all routes in $\mathcal{R}_{dt}^{\beta}$. This can be done by a modified depth-first search on the graph with the vertex set $\mathcal{L}$ in which two vertices $l_i, l_j \in \mathcal{L}$ are connected if and only if the lines l_i and l_j have a common stop. Unlike depth-first search, the algorithm considers an edge (l_i, l_j) only if l_i and l_j have at least one common stop that is served by l_i *after* the stop on which l_i was boarded. Also, if the algorithm reaches a vertex l_i that has been visited in some previous step, we need to visit all outgoing edges from l_i again because l_i was probably reached by a different edge than the edge used in the previous step. For more information, refer to Böhmová et al (2013). For the situation when one is interested in listing not only routes but also the corresponding paths in the underlying network, Böhmová et al (2014b) presented an algorithm that has a polynomial running time with respect to both the input and the output size. This algorithm traverses the network in a depth-first fashion, and for bounding the running time a stop s is visited only if there exists an st-route with sufficiently few transfers.

After listing all possible routes, we select one or more optimal route(s) $r_1, \ldots, r_k$ with one of the methods below. Furthermore we compute corresponding numbers $\gamma_1, \ldots, \gamma_k \in \mathbb{N}$ that describe how much in advance we have to leave d. If a planned timetable $T_{planned}$ is available, then we use it to find the planned journey(s) j_i along r_i that leave d as late as possible, but not later than time $t_A - \gamma_i$. Otherwise we associate the journey j_i with the route r_i, use the first possible transfer between two lines and set the departure time to $t_D^i = t_A - \gamma_i$. Thus, instead of finding robust journeys it is sufficient to compute a robust route r and the corresponding parameter γ.

A similarity-based approach. In Böhmová et al (2013), we described how a general approach to robust optimisation designed by Buhmann et al (2013) can be applied for computing robust journeys. Let $T \in \mathcal{T}$ be a timetable and $\gamma \in \mathbb{N}_0$. An *approximation set* $A_\gamma(T)$ contains all routes $r \in \mathcal{R}_{st}^{\beta}$ for which

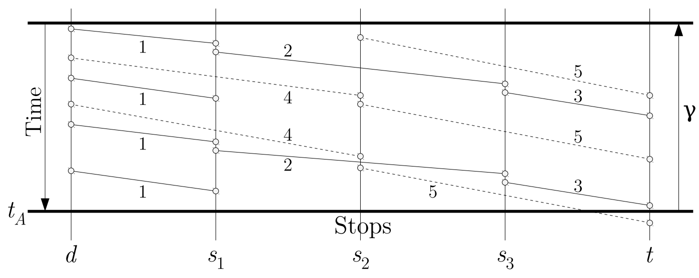

Fig. 1 A timetable with five lines $\{1,\dots,5\}$ and two routes $r_1 = \langle 1,2,3\rangle$ (solid) and $r_2 = \langle 4,5\rangle$ (dotted). The x-axis illustrates the stops $\{d, s_1, s_2, s_3, t\}$, the y-axis the time. If a trip leaves a stop s_l at time t_l and arrives at a stop s_a at time t_a, it is indicated by a line segment from (s_l, t_l) to (s_a, t_a). $A_\gamma(T)$ contains r_1 three times and r_2 once.

there exists a journey along r that departs from s at time $t_A - \gamma$ or later, and that arrives at t at time t_A or earlier (both times refer to the timetable T). We assume that $A_\gamma(T)$ is a *multiset*: a route r is contained as often as it is realised by a journey starting at time $t_A - \gamma$ or later, and arriving at time t_A or earlier (see Figure 1 for an example). The parameter γ can be interpreted as the maximal time that we depart before t_A. If we consider approximation sets $A_\gamma(T_1), \dots, A_\gamma(T_k)$ for some timetables $T_1, \dots, T_k \in \mathcal{T}$, every approximation set contains only routes that are realised in the same time period $[t_A - \gamma, t_A]$, and that are therefore comparable among different approximation sets.

The approach of Buhmann et al (2013) expects that exactly two timetables $T_1, T_2 \in \mathcal{T}$ are given. To compute a *robust* route when only two timetables are available, we consider $A_\gamma(T_1) \cap A_\gamma(T_2)$: the only chance to find a route that is likely to be good in the future is a route that was good in the past for *both* recorded timetables. The parameter γ determines the size of the intersection: if γ is too small, the intersection will be empty. If γ is too large, the intersection contains many (and maybe all) st-routes, and not all of them will be a good choice. Assuming that we knew the “optimal” parameter γ_{OPT}, we could pick a route from $A_{\gamma_{OPT}}(T_1) \cap A_{\gamma_{OPT}}(T_2)$. Buhmann et al (2013) suggest to set γ_{OPT} to the value γ that maximises the so-called *similarity* of the timetables T_1 and T_2 at value γ,

$$S_\gamma = \frac{|\mathcal{R}_{st}^{\beta}||A_\gamma(T_1) \cap A_\gamma(T_2)|}{|A_\gamma(T_1)||A_\gamma(T_2)|}. \tag{1}$$

After computing γ_{OPT} and picking a route r from the intersection, we use the planned timetable $T_{planned}$ to find a journey along r that is scheduled to leave s not later than $t_A - \gamma_{OPT}$, and use this journey as a recommendation to the user. For more details, refer to Buhmann et al (2013); Böhmová et al (2013).

Other methods. A straightforward method for finding robust routes is to enforce an additional buffer time at each transfer or at the end of the journey. The method in the previous paragraph is able to learn from past observations, but it considers only two recorded timetables. In practical applications

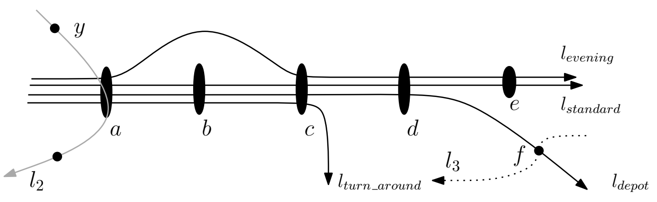

Fig. 2 Different trajectories of one line.

usually more recordings are available. If the timetables $T_1, \ldots, T_k$ are given, a straightforward idea to generalise the method from the previous paragraph is to compute the smallest γ for which $\cap_{i=1}^{k} A_\gamma(T_i) \neq \emptyset$ and select a route from the intersection of the approximation sets. A different method based on stochasticity is the following: for every route r, we compute the mean and the standard deviation of the departure times of r in every timetable T_i, and then select a route that minimises the sum of mean and standard deviation. For more details on these methods (among others), see Böhmová et al (2014a).

4 Modelling Challenges

When modelling an urban public transportation network and its behaviour, we try to capture the properties specific to this domain. However, to make a model reasonably simple and clear, we assume certain behaviour (e.g., the FIFO property) of the vehicles. While such assumptions mostly hold, in reality certain events can cause them to be violated and one has to decide whether and how to model these situations. In the following we describe some challenges that arise when dealing with real-world data (from Zürich).

Defining the lines is a crucial and a nontrivial task. The available data contain the set of stops, the planned timetable capturing the planned trips, and a detailed information on the set of trajectories for each physical vehicle during each of the days. The trajectories of physical vehicles are grouped under different labels that are used as indicators for the passengers (the label itself does not determinate the direction, nor the exact trajectory). However, a problem arises: Not all the trajectories with the same label correspond to the same sequence of stops. To illustrate this, imagine trajectories grouped under the label l as in Figure 2. Most of the day, l serves the stops $\langle a, b, c, d, e \rangle$. Twice a day it goes into the depot and serves $\langle a, b, c, d, f \rangle$, instead. In the evening, it skips b and serves only $\langle a, c, d, e \rangle$. Once a vehicle turned around in advance due to large delays and only the stops $\langle a, b, c \rangle$ were served (not planned). Even though the user may perceive these sequences as variants of the same line l, it is not clear whether to treat these lines together as one line or separately. For instance, to travel from a to f, a trajectory labelled l is recommendable only when l goes to the depot. In our model, we capture the provided information in the following way. The set of stops directly corresponds to the given set of stops. We define one line for every sequence of stops that occurs as a trajectory in the (planned) reality (e.g., l from the example is modelled by 3 different lines). Note that we define the lines to be directed, so that we can capture the situations when the "backward" line differs, e.g., due to one way streets. Clearly, even though this is a viable choice, it also has drawbacks. In the following, we give some examples of real-world situations that influence the lines and their consequences for the model and robustness.

4.1 Behaviour planned in the timetable

Standard realisation of a line. In most cases, a line $l = \langle v_1, \ldots, v_k \rangle$ is realised throughout the whole day with high frequency, and there exists a similar line (a backward line) in the opposite direction $l' = \langle v_k, \ldots, v_1 \rangle$ which contains the same stops as line l but in reverse order.

Standard realisation changes over the day. In most of the cities, there are observable patterns of how people use public transportation. During the work days, there are clearly visible peak hours, when people commute to or from work. The planned schedule of public transportation services usually tries to react to increased or decreased demand. As a consequence, the planned frequency of a line may change during the day. Sometimes, the standard realisation of the line may change during the day as well, e.g., in the evening realisation of the line, certain stops are left out completely (in Figure 2, the line $l_{evening}$ exhibits such a behaviour). As said, we model the different realisations as different lines. Usually, these lines have high frequencies during some hours, but they may not run at all during the rest of the day. This clearly brings difficulties when considering robustness issues. For example, trying to catch, but missing a last vehicle (for that day) of a particular realisation of the line may result in a significant delay of a commuter.

A vehicle goes to the depot. Roughly twice a day, a vehicle serves a line l_{depot} which partially corresponds to the line l, but differs at the end. The low frequency of such a line is highly inconvenient for the robustness considerations. Imagine a situation as in Figure 2. There may be multiple ways how to travel from y to f, however, the only possibility to get from y to f with at most one transfer might be to take the line l_2 and switch to l_{depot}. If one travels precisely at the time when l_{depot} runs and this journey worked well on the previous days, it may be reasonable to recommend him this journey. However, if l_2 is then delayed, and the transfer to l_{depot} fails, one cannot continue the recommended journey until the next line l_{depot} comes, few hours later (or even only on the next day). It is not clear whether these journeys containing low frequency lines should be recommended or not. On one hand, if in the past observations they performed well, they should not be suppressed. On the other hand, the failure mode is very inconvenient.

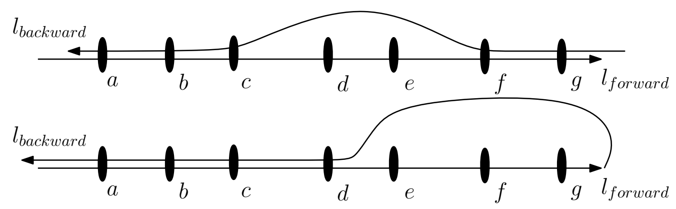

Fig. 3 The sequences of the forward and backward lines differ.

The sequence of stops of forward and backward line differs. Often, for realisation of a line there is a "backward" line that serves the exact same stops, but in the opposite order. However, in reality sometimes the backward lines differ significantly. Such a situation is often caused by one way streets, and for instance this can be widely observed in the bus network of Barcelona. Imagine a situation as in Figure 3. To get from e to c, it may be necessary to transfer from the line $l_{forward}$ to $l_{backward}$, operating under the same label. In the first case, the two lines are served by two different vehicles, and thus to transfer from one to another, one must leave the vehicle. However, in the second case, the two lines are in reality probably served by the same vehicle, that immediately after serving $l_{forward}$ continues on $l_{backward}$. Then, to transfer to $l_{backward}$, one should simply stay in the vehicle. These situations need to be modelled explicitly.

The line l in fact forms a cycle. In some rare cases, the line l does not have a corresponding backward line going in the opposite direction. Instead, after the last stop of the line l in one realisation, the vehicle which realised it continues to the first stop of the next realisation of l, thus having a cyclic topology. This is a situation that is rather difficult to capture, since in the data there is some stop that is used as an origin of the line and the cycle is broken accordingly into subsequent realisations of the line. That is, in reality, it might be possible to continue from the last stop of one realisation of the line l to the first stop of the next realisation of l, without a transfer (and without the minimum transfer time needed). This can be modelled explicitly by gluing the realisations together.

4.2 Behaviour not planned in the timetable

In reality, vehicles can always be delayed. If the delays are huge, it sometimes happens that while waiting for a vehicle of line l, no vehicle arrives for a long time and then suddenly many vehicles $V_1, V_2, \ldots$ arrive in short time intervals. Once such a situation is established, it is even intensified: since the times to pick up and drop off passengers increase (due a larger amount of waiting passengers), also the delay of V_1 and possibly also of some of the succeeding vehicles increase. This may lead to the following situations.

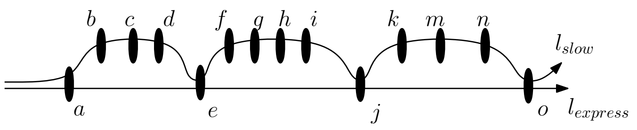

Fig. 4 An example where using one line twice is reasonable. To travel from b to m, it may be faster to travel using l_{slow} up to e, change to (a faster line) $l_{express}$ and travel to j, and finally change back to l_{slow} in order to reach m.

A vehicle turns back in advance. To avoid further delay or a dead end, the network operator may decide that a vehicle should turn around in advance before reaching the final destination. This results in a line that is realised just partially (in Figure 2, the line l_{turn_around} is an example). When such a situation occurs in the recorded timetables, we can simply set the arrival times at the stops that are *not* served by the corresponding trip to ∞. Thus, it is still possible to use such a trip as long as we do not visit stops that are not served any more. Otherwise, we simply use the next trip of the line which one would also do in reality.

One vehicle overtakes another. In some rare cases, a vehicle is greatly delayed and the succeeding vehicle of the same line l overtakes it. This usually happens only with buses. Such a situation violates the FIFO property, because there exists stops d, t on l such that taking a later dt-trip results in an earlier arrival at t. Since we assume that we always take the first arriving vehicle, it might be that the recommended journey is not optimal. However, this is not a problem from the user's perspective because in reality passengers have no knowledge that such a situation will occur and will therefore simply board the first arriving vehicle (instead of waiting for the next one).

4.3 Other Considerations

Repeated use of a single line. As we discussed earlier, it may be necessary to use multiple lines of the same label for a single journey (e.g., when the forward and backward lines differ). In fact, we may want to use a single line repeatedly, as Figure 4 shows: Imagine that there is an express line $l_{express}$ with very few stops, that stops only in significant places of the city and there is also a slow line l_{slow}, that serves a similar trajectory, but stops more often. Then, to get from an "unimportant" stop b to an "unimportant" point m, one possibility is to take the slow line for the entire journey. However, it may be more efficient to start with the slow line, and as soon as it is possible (at stop e), change to the express line and then change back to the slow line (at j) only to reach stop m (which is not on the express line). On the other hand, every transfer from a line to another increases the risk of missing the next connection due to delays in the network. Thus, it is an interesting problem, to detect and decide whether or not the repeated use of the same line or label is justified, necessary, or desired.

However, certain observations can be made. In particular, transferring from a line l to the very same line immediately afterwards, can hardly result in any benefit. Similarly, solutions containing immediate transfers to the same line in the opposite direction (that serves the exact same stops, but in the opposite order) can also be discarded.

Nearby stations. Some of the larger stations, where many lines meet, are sometimes split into several stops that are very close to each other, but have slightly different names. The question is whether these stops should be considered together as one station, or not. The argument towards joining would be the fact that one may want to use such compound station as a transfer station. An argument against would be that even though the stops are relatively close to each other, it may not be easy to find them for a person who does not know the area. Thus, it is not clear how much time would one need on such a transfer. Multimodal solution (adding walking arcs) may help to solve the problem, however, it remains difficult to set the minimum transfer times so that the non-local people have enough time to find the right stop, but without creating too much slack time for the people familiar with the area. Furthermore, people with reduced mobility may prefer to avoid transfers between the stops in a compound station completely.

5 Conclusion

We discussed a model for urban public transportation networks that facilitates the search for robust journeys. We described some of the challenges that arise from real-world network and delay information, and we discussed their impact on the model.

The fact that some of the trajectories are realised only very rarely has negative consequences on the robustness. We believe that unifying the lines on parts where they serve the same sequence of stops, would lead to a better overall frequencies of the lines and thus to more robust solutions. Then, the recommendation to the user could be in the form “First, take l_1 or l_5 in the direction D and change to l_4 at stop a.” It is not clear how to do this algorithmically in a concise and systematic way, and we plan to consider this issue in a further research.

References

Bast H, Carlsson E, Eigenwillig A, Geisberger R, Harrelson C, Raychev V, Viger F (2010) Fast routing in very large public transportation networks using transfer patterns. In: ESA, pp 290–301

Bauer R, Delling D, Wagner D (2011) Experimental study of speed up techniques for timetable information systems. Networks 57(1):38–52

Böhmová K, Mihalák M, Pröger T, Srámek R, Widmayer P (2013) Robust routing in urban public transportation: How to find reliable journeys based on past observations. In: ATMOS, pp 27–41

Böhmová K, Mihalák M, Neubert P, Pröger T (2014a) Robust routing in urban public transportation: Evaluating strategies that learn from the past. Tech. Rep. eCompass-TR-057, eCompass

Böhmová K, Mihalák M, Pröger T, Sacomoto G, Sagot MF (2014b) Computing and listing st-paths in public transportation networks. Tech. Rep. eCompass-TR-056, eCompass

Boyan J, Mitzenmacher M (2001) Improved results for route planning in stochastic transportation. In: SODA, pp 895–902

Brodal GS, Jacob R (2004) Time-dependent networks as models to achieve fast exact time-table queries. Electronic Notes in Theoretical Computer Science 92:3–15

Buhmann JM, Mihalák M, Srámek R, Widmayer P (2013) Robust optimization in the presence of uncertainty. In: ITCS, pp 505–514

Delling D, Pajor T, Wagner D (2009) Engineering time-expanded graphs for faster timetable information. In: Robust and Online Large-Scale Optimization, Springer, pp 182–206

Delling D, Pajor T, Werneck RF (2012) Round-based public transit routing. In: ALENEX, pp 130–140

Dibbelt J, Pajor T, Strasser B, Wagner D (2013) Intriguingly simple and fast transit routing. In: SEA, pp 43–54

Disser Y, Müller-Hannemann M, Schnee M (2008) Multi-criteria shortest paths in time-dependent train networks. In: WEA, pp 347–361

Firmani D, Italiano GF, Laura L, Santaroni F, et al (2013) Is timetabling routing always reliable for public transportl. In: ATMOS, pp 15–26

Frank H (1969) Shortest paths in probabilistic graphs. Operations Research 17(4):583–599

Goerigk M, Knoth M, Müller-Hannemann M, Schmidt M, Schöbel A (2011) The price of robustness in timetable information. In: ATMOS, pp 76–87

Müller-Hannemann M, Schnee M (2009) Efficient timetable information in the presence of delays. In: Robust and Online Large-Scale Optimization, Springer, pp 249–272

Müller-Hannemann M, Weihe K (2001) Pareto shortest paths is often feasible in practice. In: WAE, pp 185–197

Müller-Hannemann M, Schulz F, Wagner D, Zaroliagis C (2007) Timetable information: Models and algorithms. In: Algorithmic Methods for Railway Optimization, Springer, pp 67–90

Nachtigall K (1995) Time depending shortest-path problems with applications to railway networks. European Journal of Operational Research 83(1):154–166

Nikolova E, Kelner JA, Brand M, Mitzenmacher M (2006) Stochastic shortest paths via quasi-convex maximization. In: ESA, pp 552–563

Orda A, Rom R (1990) Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM 37(3):607–625

Orda A, Rom R (1991) Minimum weight paths in time-dependent networks. Networks 21(3):295–319

Pallottino S, Scutella MG (1998) Shortest path algorithms in transportation models: classical and innovative aspects. In: Equilibrium and advanced transportation modelling, vol 245, Springer, p 281

Pyrga E, Schulz F, Wagner D, Zaroliagis C (2008) Efficient models for timetable information in public transportation systems. ACM Journal of Experimental Algorithmics 12:2–4

Schulz F, Wagner D, Zaroliagis C (2002) Using multi-level graphs for timetable information in railway systems. In: ALENEX, pp 43–59