Better Transit Routing by Exploiting Vehicle GPS Data (Research Paper)[*]

Daniel Delling
Microsoft Research
dadellin@microsoft.com

Giuseppe F. Italiano
Università di Roma
"Tor Vergata"
giuseppe.italiano@uniroma2.it

Thomas Pajor
Microsoft Research
tpajor@microsoft.com

Federico Santaroni
Università di Roma
"Tor Vergata"
santaroni@ing.uniroma2.it

ABSTRACT

Current public transport journey planners are mostly based on timetables, i.e., their planning assumes implicitly that all transit vehicles run on schedule. Unfortunately, unpredictable delays occur frequently which may have a negative impact on the quality of the journeys provided by timetable-based planners. In this paper, we check solutions for mitigating this problem and evaluate them empirically on the metropolitan public transportation network of Rome (Italy). We first try to assess whether the availability of dynamic information on the geo-location of transit vehicles (via GPS data) may help to improve the quality of the journeys offered by a planner. The main findings of our experiments are that GSP data provides substantial benefits for short journeys, while it does not seem to provide enough insight to improve long journeys. In particular, even with the use of GPS data, a journey planner can still be rather unreliable for long journeys, as there can be large fluctuations between the times predicted by the planner and the actual travel times incurred by passengers. As a second contribution of our work, we provide and evaluate experimentally new routing algorithms to improve the reliability of public transport journey planners.

1. INTRODUCTION

We consider the problem of finding journeys in metropolitan schedule-based public transit networks where delays occur. Recently, some cities have equipped all buses with GPS devices to collect their current positions in real time. Leveraging this *GPS feed*, it enables the transit agency to have an almost complete overview of the current traffic situation throughout the city. While some agencies expose this data to the user (e.g., via their websites to warn about potentially missed connections), it is usually not used within the journey planner itself, with the result that recommended journeys may lead to missed transfers and late arrivals.

[*]The work of the second and the fourth authors has been partially supported by the Italian Ministry of Education, University and Research, under Project AMANDA (Algorithmics for MAssive and Networked DAta).

In this paper we study how to improve the quality of a journey planning system by taking GPS feeds into account in the *planning stage*. We first check the quality of journeys computed according to the timetable (which ignores delays). We then show how to incorporate the real-time GPS feed of the current traffic situation into the timetable (also predicting the near future) and evaluate the effect on journey quality. We also develop a novel extension for RAPTOR (a state-of-the-art journey planning algorithm [9]), making the computed journeys more robust to delays. Our extensive experimental study on the *real* GPS feed from Rome, Italy reveals that we are able to improve the quality of the planned journeys.

To the best of our knowledge, we are among the first to use GPS data to improve a public transport journey planning system. There has been work on efficient algorithms for journey planning [2, 9, 11, 20] (see [3] for a survey). Our findings in this work translate to other journey planning systems easily, as long as they do not rely on heavy preprocessing. The robustness to delays of journeys has been studied in [4, 13], with the obvious result: As long as the delays are minor, computing journeys according to the timetable is sufficiently good. However, preliminary experiments with productions systems such as Google Transit (`www.google.com/transit`) or Muovi Roma (`http://www.agenziamobilita.roma.it`) indicate that, when deviations from the schedule become significant, incorporating delays would indeed benefit the quality of the computed journeys [1].

Making journey planning algorithms more robust to heavy delays has been studied before [7,12,14,15,18,19], but none of these papers has considered GPS feeds. An important building block for our work is to predict the near future from the current traffic situation. Some previous work [6,19] focused only on how delays propagate through a train network if one train is delayed and others are waiting to avoid missed connections. Previous work with GPS data [5, 16, 17, 21] focused on improving the prediction of travel times in metropolitan transport networks, but did not evaluate whether it can improve a journey planner for public transportation.

2. PRELIMINARIES

We work with aperiodic *timetables*, which consist of a set $\mathcal{S}$ of *stops*, a set $\mathcal{R}$ of *routes*, a set $\mathcal{T}$ of *trips*, and a set $\mathcal{F}$ of *footpaths*. Each stop $p \in \mathcal{S}$ represents a distinct location at which one can board or exit a vehicle (such as a bus stop or a subway platform). Each trip $t \in \mathcal{T}$ is a sequence of stops a particular vehicle follows during the day. Every stop $p \in t$ along the trip has an associated arrival and departure time $\tau_{\text{arr}}(t,p)$ and $\tau_{\text{dep}}(t,p)$, where $\tau_{\text{arr}}(t,p) \leq$

$\tau_{\text{dep}}(t,p)$. Trips that follow the same sequence of stops without overtaking each other are grouped into routes $r \in \mathscr{R}$, ordered by their departure time at the first stop. Finally, each footpath $f = (p_1, p_2, \ell) \in \mathscr{F}$ enables walking from stop p_1 to stop p_2 in time ℓ.

The output of a journey planning algorithm is a set of *journeys* $\mathscr{J}$. A journey $J \in \mathscr{J}$ is a sequence of trips and footpaths in order of travel. Every trip $t \in J$ has associated stops for pick-up and drop-off with their corresponding departure and arrival times. A journey with k trips has $k-1$ *transfers*, and its *travel time* $\tau_{\text{trav}}(J)$ is the difference $\tau_{\text{arr}}(J) - \tau_{\text{dep}}(J)$ of its arrival and departure times. Moreover, we assign each journey several *criteria* $x(J)$, such as its *arrival time* and *number of transfers*. A journey J_1 *dominates* a journey J_2, if J_1 is better (or equal) in all criteria than journey J_2. Given source and target stops p_s and p_t, a departure time τ_{dep} and several optimization criteria, we are interested in computing a *Pareto set* of journeys $\mathscr{J}$ optimizing the given criteria. More precisely, each journey $J \in \mathscr{J}$ must leave p_s no earlier than τ_{dep}, terminate at p_t, and no two journeys $J_1, J_2 \in \mathscr{J}$ may dominate each other.

The recent *RAPTOR* algorithm [9] efficiently computes Pareto sets of journeys optimizing arrival time and number of transfers. It works in rounds, computing in round i earliest arrival times for all stops that can be reached with i trips ($i-1$ transfers). It maintains for each stop p and round i a *label* $\tau_i(p)$ (initially set to ∞), the earliest arrival time to get to p with i trips. The algorithm initializes $\tau_0(p_s) = \tau$ (recall that τ is the departure time at the source stop p_s, given as input). For each round i, beginning with $i = 1$, it first copies all labels $\tau_i(\cdot) = \tau_{i-1}(\cdot)$, and then *scans* all routes hit by a stop whose label has been *updated* in the previous round (for the first round this is only p_s). To scan a route r, RAPTOR maintains a tentative trip t (which is initially undefined) traversing the stops of the route in order. At every stop p, it first checks whether $\tau_{\text{arr}}(t,p) < \tau_i(p)$, in which case it sets $\tau_{\text{arr}}(t,p) = \tau_i(p)$ (think of this as leaving trip t at stop p). It then attempts to update the tentative trip: if $\tau_{i-1}(p) < \tau_{\text{dep}}(t,p)$, it sets t to be the earliest possible trip of route r that departs p after $\tau_{i-1}(p)$ (think of this as boarding trip t at p). The algorithm terminates if no labels have been updated in a round, resulting in a *provably exact* Pareto set at every stop. In case one is only interested in a target stop p_t, *target pruning* accelerates the algorithm by pruning journeys (anywhere) in the network that provably do not contribute to the Pareto set at p_t.

RAPTOR can be extended to McRAPTOR [9] to incorporate further optimization criteria (beyond arrival time and number of transfers). It still works in rounds, but it maintains with each stop p and round i a *bag* $B_i(p)$ of nondominated labels (journeys). (Besides arrival time, these label have associated values for each further criterion.) To scan a route r, McRAPTOR must maintain a *tentative bag* B of labels with associated trips. At each stop p along the route, it first *merges* B into $B_i(p)$: it updates all affected criteria (e. g., the arrival time) of the labels of B and copies them into $B_i(p)$, removing dominated labels on the fly. Similarly, it then merges $B_{i-1}(p)$ into B, assigning (earliest trips) to each newly-added label of B. The algorithm terminates after round i if no new labels have been added to any bag $B_i(\cdot)$ in round i, resulting in exact Pareto sets of journeys at each stop. If one is only interested in journeys to a target stop p_t, target pruning can be extended to McRAPTOR, as well.

3. METHODOLOGY

In this section we describe how we collect GPS data, how we evaluate the quality of a journey with it, and our experimental setup.

We collected GPS data from transit agency of Rome. For each bus in the network, we gathered the route on which it is operating, the relative position of the bus in the considered route, and the time of the last GPS update for the given bus. If the GPS stream was incomplete for a bus (due to connection loss) we simply interpolated.

Our simulation system is able to simulate the experience of a user traveling according to the solution reported by a journey planner. Analyzing the GPS data, it computes the *actual* arrival time of the user at their destination. From this we compute the travel time $\tau_{\text{trav}}(J)$ of the journey J as the difference between the actual arrival time and the time at which the query was issued. Note that we collect GPS data only for buses, and thus assume that subways operate according to the timetable (a reasonable assumption).

We define the *prediction error* of a journey J as the absolute value of the relative error of the time estimates, i.e., $|\tilde{\tau}_{\text{trav}}(J) - \tau_{\text{trav}}(J)| / \tau_{\text{trav}}(J)$, where $\tilde{\tau}_{\text{trav}}(J)$ and $\tau_{\text{trav}}(J)$ are, respectively, the estimated and the actual travel times of the journey J. We consider the absolute value of the relative error, since we are interested in evaluating how far the time estimates are from the actual times, independently of the fact that the error is due to optimistic or pessimistic predictions. If the estimated arrival time is exactly the same as the actual arrival time, the quality measure is 0.

We define the *best journey time* as the actual time required by the best journey J^* answering q, computed taking into account the complete GPS stream of the full day *in advance*. This is a particularly important parameter, since it provides a lower bound for the quality of computed journeys. We define the *distance from the best* of a journey J as $\tau_{\text{trav}}(J) - \tau_{\text{trav}}(J^*)$. If the considered journey is the fastest solution, this value is 0.

We consider the public transport network of the metropolitan area of Rome. It consists of 309 bus lines and 2 subway lines, involving a total of 7,089 stops (7,037 bus stops and 52 subway stops). We implemented all algorithms in C++, compiled with GCC 4.5, and evaluated them on an Intel Core i7 CPU. We compute a set of 5,000 queries with origin and destination stops chosen uniformly at random. We choose departure times uniformly at random in the time range 7:30 am to 7 pm. We use the GPS data of Wednesday, February 26, 2014, a typical day with no unusual traffic jams, construction works or extreme weather conditions.

We then run the same queries with all considered algorithms and collect the returned journeys together with their time estimates. For each such journey, we use our simulation system to compute its actual time, computing the quality measures from above. Given a set of journeys, we compare the performances of different algorithms by the median, 10th and 90th percentile for each quality measure. As we consider the lowest and the highest 10% of values as outliers, we can focus on the most reliable 80% of the measurements. We distinguish between short (less than 30 minutes), mid (30 to 60) and long (longer than 60) range journeys.

4. IGNORING THE GPS FEED

First, we evaluate the impact of traffic on journey quality, if they are obtained without leveraging GPS data. For this, we run RAPTOR on the normal timetable, and only use our GPS data to evaluate the journey quality *after* computing them. We first consider plain RAPTOR, which we call *RAPTOR(TT)* to make clear that it runs on the original timetable. RAPTOR returns a set of journeys from which we pick the one with earliest arrival time.

Figure 1 (left) illustrates the prediction error of RAPTOR(TT) as a function of the journey time estimated by the algorithm. In this plot, the x-axis is divided into time slots of 5 minutes: for each time slot, we report the value of the prediction error for all the journeys falling in that range, together with their median, 10th and 90th percentile. We observe that in 50% of the cases, the time estimates computed by RAPTOR(TT) can be affected by a considerable prediction error, which is more than 20% for mid and long range journeys, and of the order of 30% for small range journeys. Higher relative errors for

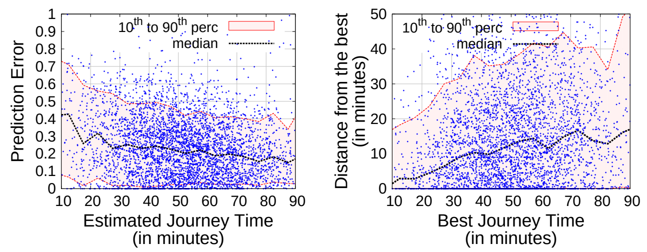

Figure 1: Prediction error (left) and distance from the best (right) of journeys computed by RAPTOR(TT).

short range journeys can be explained by the fact that even small delays can yield relatively large fluctuations in the prediction error.

Figure 1 (right) illustrates the distance from the best of the same journeys, as a function of the best journey time. As we see, in several cases the solutions provided by RAPTOR(TT) can be substantially far away from the best possible solution.

Next, we check whether we can improve the route quality by optimizing *transfer reliability* [12]. The reliability of a transfer from trip t_1 to trip t_2 is a function (called rel) of the transfer's *buffer time* (i. e., $\tau_{\text{dep}}(t_2, p) - \tau_{\text{arr}}(t_1, p)$) to the interval $[0,1]$. (Note that the buffer time indicates by how much t_1 may be delayed at p before t_2 is missed.) We use an exponential function to model reliability, that is, $\text{rel}(\tau) = e^{\ln(1-a)-b/\tau}$. In our experiments we set a and b such that $\text{rel}(0\,\text{min}) = 0.5$ and $\text{rel}(10\,\text{min}) = 0.99$. For better performance, we subdivide the codomain of rel into 100 equivalence classes of equal width. The reliability of a journey J is now the product over the reliabilities of each individual transfer in J.

We use the *RcRAPTOR* algorithm (Reliability Criterion RAPTOR) to compute Pareto sets optimizing arrival time, number of transfers, and reliability [10]. It is essentially identical to McRAPTOR (using pairs of arrival time and reliability as labels), however, with one distinction: Whenever it attempts to merge a bag $B_{i-1}(p)$ into the tentative route bag B (while scanning route r at stop p), it may create multiple new nondominated labels with associated trips for every label $L \in B_{i-1}(p)$. Besides the earliest trip departing after $\tau_{\text{arr}}(L)$, RcRAPTOR must consider all subsequent (later) trips that may provide a higher reliability (at the cost of arriving later). The algorithm produces a Pareto set of journeys that differently trade reliability for arrival time (besides number of transfers). For the following study, we restrict ourselves to picking one journey (per round i) from the Pareto set a posteriori. *RcRAPTOR-MR* picks the journey with *maximum* reliability while *RcRAPTOR-AVG* picks the one whose reliability is closest to the *average* reliability (of all journeys computed).

Figure 2 (left) shows that taking the reliability of transfers explicitly into account (as RcRAPTOR does) seems a crucial asset for computing more reliable time estimates. Indeed, both variants of

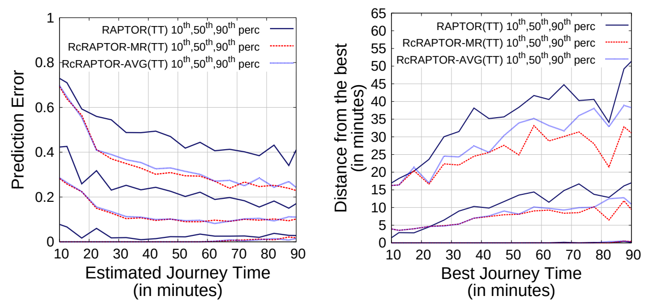

Figure 2: Prediction error (left) and distance from the best (right) of journeys computed by RAPTOR(TT), RcRAPTOR-MR(TT) and RcRAPTOR-AVG(TT).

RcRAPTOR significantly outperform RAPTOR(TT) in terms of the accuracy of time estimates (prediction error).

Figure 2 (right) illustrates the distance from the best for the same journeys. Beside the general improvement of the RcRAPTOR algorithms over RAPTOR, two points are interesting. Firstly, for very short journeys RAPTOR(TT) appears to be closer to the optimum than either RcRAPTOR-MR(TT) or RcRAPTOR-AVG(TT). This is due to the fact that RcRAPTOR optimizes reliability, and hence prefers longer journeys with larger (thus, more reliable) waiting times for transfers. In this case, the impact of longer waiting times might be more noticeable on very short journeys. Note that for mid to long range journeys, which tend to involve more transfers, the reliability introduced by the RcRAPTOR variants produce a significant improvement over RAPTOR(TT).

Secondly, Figure 2 (right) indicates that RcRAPTOR-MR(TT) is closer to the optimum than RcRAPTOR-AVG(TT), especially for mid and long range journeys. As previously observed, those journeys involve more transfers, and missing one transfer introduces delays in the journey. In this case, always selecting the journey with maximum reliability pays off.

5. USING THE GPS FEED

The experiments from the previous section show that the reliable approach taken by RcRAPTOR over basic RAPTOR introduces effective protection against delays and other deviations from the original schedule. It seems natural to ask whether one could obtain even further improvements by taking explicitly into account the dynamic information about the geographic location of transit vehicles, available through GPS data. This is explored in this section.

5.1 Traffic Prediction

Algorithms in the RAPTOR family, like most of the public transport routing algorithms, expect their input in form of a timetable, which completely describes future departure and arrival events of all vehicles. In contrast, GPS data provides very detailed information about past events, but gives very little information about the future. Namely, let τ be a given time at which a query is issued to the algorithm. The GPS data indicates how all transit vehicles moved in the network before or at time τ, but in order to answer the query we also need to predict the positions of the vehicles after time τ.

Since this is not the main focus of our work, we consider a very simple predictive model in our experiments. In particular, we assume that the average speed of transit vehicles on a given route observed before time τ provides a good approximation of the future speed of vehicles on the very same route. More precisely, we consider the speed of vehicles along each *hop* $h = (p_i, p_{i+1})$ of its trip, i.e., between stop p_i and its consecutive stop p_{i+1}. Whenever we receive new GPS data, we update the (average) speed for each affected hop. By these means, we can predict the arrival times for each vehicle at future stops. For trips (vehicles) which are not yet in operation (at the current time τ), we assume that they will depart from their first stop according to the timetable and will proceed according to current speed estimations.

5.2 RAPTOR

To assess the impact of GPS data on the journeys computed by RAPTOR-based algorithms, we start our experiments by comparing RAPTOR(TT) and RAPTOR(GPS) (i.e., RAPTOR running on GPS data). In particular, Figure 3 (left) illustrates the prediction error of the journeys computed by RAPTOR(TT) and by RAPTOR(GPS) as a function of the estimated journey time. Since GPS data seems to provide more reliable information than timetable data, one would expect that its usage could substantially improve the accuracy of

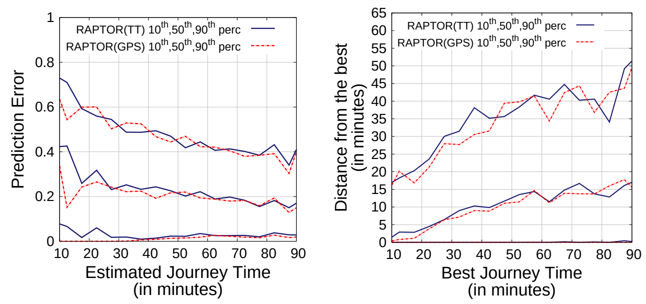

Figure 3: Prediction error (left) and distance from the best (right) of journeys computed by RAPTOR(TT) and RAPTOR(GPS).

the time estimates used by the journey planner. However, although there are improvements for short journeys, for mid and long range journeys there seems to be no substantial difference in the time estimates produced with GPS data and with timetable data. Note that our prediction model is rather simple; perhaps, one could improve the prediction error for longer-range journeys with a more sophisticated model.

Our experiments highlight that the main impact of GPS data depends on the pick-up time of the first bus: the sooner (after the time the query is issued) the first bus is taken, the more effective GPS data seems to be in improving the prediction error. This is intuitive: if the user has to catch a bus in a few minutes, then GPS data is more accurate; on the other hand, if the user is first taking a long subway trip before transferring to a bus, the predicted delays from the (current) GPS data are likely to be much less acurate.

To show this phenomenon, we present in Figure 4 a different visualization of the results of the experiment illustrated in Figure 3 (left): this time, the x-axis represents the pick-up time of the first bus along the journey computed using GPS data, and the y-axis represents the difference $\tau_{\text{trav}}(J_{\text{tt}}) - \tau_{\text{trav}}(J_{\text{GPS}})$ between the actual time of the journey computed using timetable data and the one using GPS data. Obviously, positive points in the plot (i.e., $\tau_{\text{trav}}(J_{\text{tt}}) - \tau_{\text{trav}}(J_{\text{GPS}}) > 0$) represent a positive impact of GPS data, while negative points represent a negative impact of GPS data. Queries returning the very same journey, both, for GPS and for timetable data are excluded from Figure 4, since they do not provide insight into the differences between the use of GPS and timetable data.

As we observe from Figure 4, the effectiveness of GPS data decreases consistently with the pick-up time of the first bus: in particular, we see a positive effect when the pick-up time of the first bus is at most 50 minutes along the journey, a negative effect for more than 70 minutes, and a neutral effect in between.

In summary, although with our simple prediction model, GPS data improves the accuracy of the time estimates mostly for short journeys, it can nevertheless provide better solutions than timetable data for journeys where the pick up of the first bus is not too far in the future. We thus expect that the benefits of GPS data are not confined to short range journeys only, but also carry over to mid and long range journeys with an early pick-up time of the first bus. Indeed, as illustrated in Figure 3 (right), which reports the distance from the best of the journeys computed by RAPTOR(TT) and RAPTOR(GPS), GPS data seems to provide solutions slightly closer to the optimum than timetable data, even for long-range journeys.

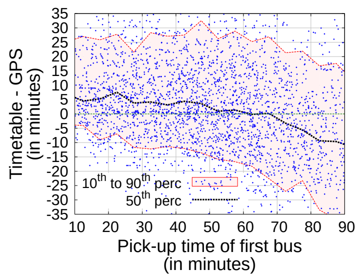

Figure 4: RAPTOR(TT) compared to RAPTOR(GPS). The difference between the actual times of journeys computed using both timetable and GPS data is plotted as a function of the pick-up time of the first bus (better viewed in color).

5.3 Reliablity

Next, we again consider reliability. Namely, we compare RAPTOR(GPS), RcRAPTOR-AVG(GPS) and RcRAPTOR-MR(GPS) to evaluate the impact of optimizing reliability on the basis of GPS data. Figure 5 reports the results analogously to Figure 2, but now for GPS data. As for the case of timetable data, the RcRAPTOR variants with GPS data provide substantial improvements over RAPTOR(GPS): taking reliability of transfers explicitly into account is beneficial for computing more reliable time estimates and for obtaining solutions that are closer to the optimum. We note that the usage of GPS data seems to mitigate one issue that we mentioned for the experiment reported in Figure 2 (right). Differently from timetable data, for very short journeys now RcRAPTOR-MR(GPS) and RcRAPTOR-AVG(GPS) appear to be closer to the optimum than RAPTOR(GPS). Using GPS data seems to compensate the previously mentioned conservative effect of RcRAPTOR in penalizing journeys with short waiting times.

In summary, our experiments with RAPTOR and RcRAPTOR show that GPS data is likely to be beneficial especially for short journeys, while it does not seem to provide enough insight to substantially improve the quality of long journeys. Indeed, for the case of short journeys, the use of GPS data is able to, both, increase the accuracy of the time estimates, and decrease the distance of the solutions provided from the optimum. In contrast, for long journeys the use of GPS data does not seem to offer any substantial advantage over timetable data (both in terms of prediction error and distance from the best). However, RcRAPTOR seems to produce improved solutions over RAPTOR (for timetable data as well as GPS data), especially for mid and long range journeys. In particular, long journeys obtained by RcRAPTOR-MR(GPS) or RcRAPTOR-AVG(GPS) take on average about 10% longer than their original estimate, while the same journeys obtained by RAPTOR(GPS) take an average of 20% longer than their original estimate. Further improving the reliability of the solutions provided by a journey planner may be particularly important, as some users might be willing to trade time for greater reliability, i.e, they might prefer to complete their journey in slightly more time but with a stronger guarantee to reach the final destination by a given deadline.

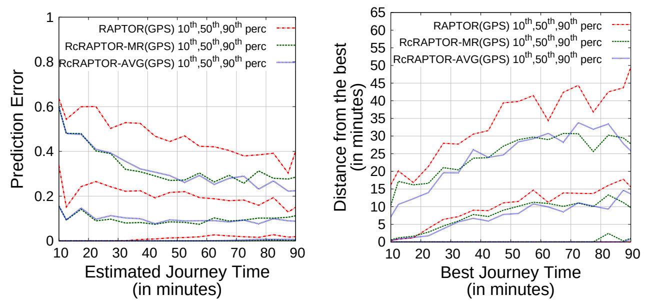

Figure 5: Prediction error (left) and distance from the best (right) of journeys computed by RAPTOR(GPS), RcRAPTOR-MR(GPS) and RcRAPTOR-AVG(GPS).

6. INCREASING ROBUSTNESS

Motivated by the discussion above, we now introduce a new RAPTOR-based algorithm, which aims to reduce the prediction error, i.e., to improve the accuracy of the time estimates.

We refer to the event of missing a scheduled transfer during a journey as a *fault*. Note that a fault can happen in case of a mismatch between predicted and actual times: an incoming transit vehicle may arrive too late at a given stop (with respect to its predicted arrival) or an outgoing transit vehicle may depart too early from a given stop (with respect to its predicted departure). For the sake of simplicity, we assume that users always follow the journeys offered by the planner, i.e., whenever a fault occurs, the user simply waits for the next transit vehicle on the same route and does not deviate from the journey. The occurrence of a fault may clearly cause further delays. We associate to each fault the resulting delay caused in the journey, when only that fault occurs. We refer to the fault in a journey causing the largest delay as its *worst fault*. Intuitively, we would like to have routing algorithms that are robust against worst faults, i.e., they return journeys where worst faults produce small delays. Note that those algorithms are likely to be effective in practice as long as at most one fault can happen during a journey.

6.1 Extending RAPTOR

We now propose 1-Fault-Aware RAPTOR, a new algorithm which aims at being robust against worst faults. Similarly to McRAPTOR, 1-Fault-Aware RAPTOR is based on multicriteria optimization through Pareto optimality. However, differently from RcRAPTOR, 1-Fault-Aware RAPTOR does not consider the reliability of transfers, but rather tries to minimize the largest possible delay in case of one missed transfer.

Similarly to RAPTOR, in round i it computes for every stop p the earliest arrival time at p when using at most i trips ($i-1$ transfers), denoted by $\tau_i(p)$. In addition, it explicitly considers the arrival time for the case that the worst fault just occured before stop p, denoted by $\tau_i^{\mathrm{wf}}(p)$. We now sketch the modifications required for RAPTOR. Let $q = (p_s, \tau, p_t)$ be a query where p_s is the origin stop, τ the departure time, and p_t the destination stop.

We first initialize both $\tau(p_s)$ and $\tau^{\mathrm{wf}}(p_s)$ to τ, since at time τ the journey starts from the origin. At the end of the first round, we compute the best journeys (according to Pareto optimality) to get to every reachable stop p with exactly one trip. To this end, we compute $\tau(p)$, i.e., the earliest arrival time to p if no fault occurs, and we set $\tau^{\mathrm{wf}}(p)$ to the arrival time at p if we miss the first available vehicle to p on the only route considered in the journey. For rounds $i > 1$, 1-Fault-Aware RAPTOR works as follows. If one more trip (with respect to the previous round) reaches a new stop, it works exactly as in the first iteration. Otherwise, if using one more trip departing from a stop p_0 of the computed journey leads to an already reached stop p', we compute the new solution and we update the non-dominating set of solutions leading to p' according to Pareto optimality. To this end, we need to compute both $\tau(p')$ and $\tau^{\mathrm{wf}}(p')$, i.e., the earliest arrival time to p' if, respectively, no fault or the worst fault occurs. The value $\tau(p')$ can be easily computed exactly as for basic RAPTOR (cf. Section 2). To compute $\tau^{\mathrm{wf}}(p')$, i.e., the arrival time at p' if the worst fault occurs, we use the following formula: $\tau^{\mathrm{wf}}(p') = \max\{\tau^{\mathrm{f}}(\tilde{p^0}, p'), \tau^{\mathrm{f}}(p^0, \tilde{p'})\}$, where $\tau^{\mathrm{f}}(\tilde{p^0}, p')$ and $\tau^{\mathrm{f}}(p^0, \tilde{p'})$ are, respectively, the arrival time if a fault occurred before p^0 or at p^0. Recall that p_0 is the stop from which the new transit vehicle departs during the current round, and that at most one fault can occur.

When the algorithm terminates, it obtains a set of nondominating journeys leading to p_t. Among these journeys, 1-Fault-Aware RAPTOR returns the one with the lowest difference between $\tau^{\mathrm{wf}}(p_t)$ and $\tau(p_t)$. Notice that this choice does not guarantee that we are selecting the fastest journey.

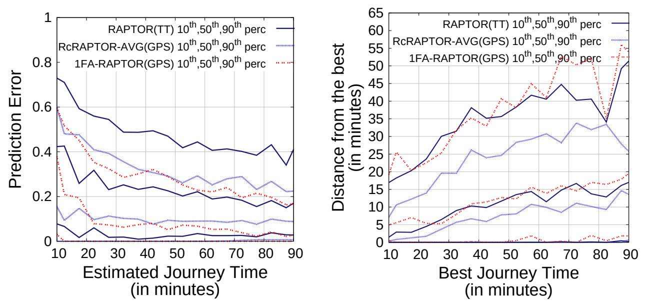

Figure 6: Prediction error (left) and distance from the best (right) of journeys computed by RcRAPTOR-AVG(GPS) and 1-Fault-Aware RAPTOR(GPS), keeping RAPTOR(TT) as our baseline.

6.2 Results

Next, we report the results from our experiments with 1-Fault-Aware RAPTOR(GPS). Compared to the other algorithms considered so far, one expects that the journeys provided by 1-Fault-Aware RAPTOR(GPS) have more reliable time estimates which could be at the same time further away from the best possible journeys. In other terms, 1-Fault-Aware RAPTOR(GPS) is expected to deliver slower but more reliable journeys, by improving the prediction error at the expense of worsening the distance from the best. Figure 6 illustrates the results of our experiments with 1-Fault-Aware RAPTOR(GPS). To put those results in perspective, we include in the same plot the results of the same experiment with the best and the worst algorithm resulting from our experiments, respectively RcRAPTOR-AVG(GPS) and RAPTOR(TT).

We first analyze the prediction error of the three algorithms, shown in Figure 6 (left). As expected, 1-Fault-Aware RAPTOR(GPS) is able to deliver more reliable solutions in case of mid to long range journeys. In particular, long journeys suggested by RcRAPTOR-AVG(GPS) take on average about 10% more than their original estimate, while the estimates on the long journeys suggested by 1-Fault-Aware RAPTOR(GPS) appear to be more reliable: indeed, in this case a journey takes an average of only about 2% more than its original estimate. We note that 1-Fault-Aware RAPTOR(GPS) does not seem to be able to provide reliable solutions for very short journeys. The main reason for this is that 1-Fault-Aware RAPTOR(GPS) is not likely to provide very short journeys: as a consequence, in our experiments with 1-Fault-Aware RAPTOR(GPS) there are only few points corresponding to journeys with estimated time less than 20 minutes, which do not seem to be statistically significant. Additionally, the same relative error appears to be more relevant in practice for long journeys rather than for short journeys: a 20% relative error on an estimated journey time of 15 minutes, yields an actual delay of 3 minutes, while a 20% relative error on an estimated journey time of 90 minutes yields an actual delay of 18 minutes.

We next analyze the distance from the best of the solutions provided by the three algorithms, as illustrated in Figure 6 (right). As expected, the improved reliability in the time estimates of 1-Fault-Aware RAPTOR(GPS) comes at the cost of a larger distance from optimality. However, the resulting performance degradation does not appear to be significant: indeed, the journeys provided by 1-Fault-Aware RAPTOR(GPS) are about only 5 minutes slower on average than the journeys provided by RcRAPTOR-AVG(GPS), and seem to be very close to the journeys provided by RAPTOR(TT).

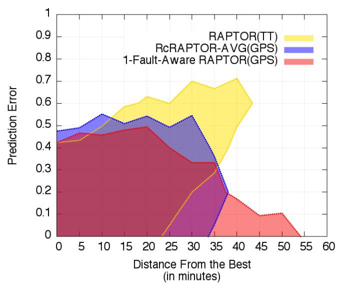

Figure 7: Distribution of the solutions computed by RAPTOR(TT), RcRAPTOR-AVG(GPS) and 1-Fault-Aware RAPTOR(GPS), with respect to both the prediction error and the distance from the best.

In summary, compared to RcRAPTOR-AVG(GPS), 1-Fault-Aware RAPTOR(GPS) seems to provide journeys with more reliable estimates (2% on average, compared to 10%) that are only 5 minutes slower on average.

Figure 7 further analyzes the tradeoffs between reliability of the estimates and distance from optimality, by plotting the distribution of journeys computed by 1-Fault-Aware RAPTOR(GPS), compared to RcRAPTOR-AVG(GPS) and RAPTOR(TT). The *x*-axis represents the distance from the best, while the *y*-axis represents the prediction error: each journey is thus represented by a point on this plane. We divided the plane into small regions and considered, for each algorithm, the number of journeys falling into each region. We discarded regions with very low density of points, until we removed about 10% of them, which were considered as outliers. The remaining 90% of the points highlight contiguous regions describing, for each algorithm, in which part of the plane the solutions are likely to fall with high probability.

In theory, one would like to achieve both a small distance from the best and a small prediction error, so an ideal algorithm would concentrate its solutions around the origin of this plane. Points close to the *x*-axis represent journeys that are robust but not necessarily fast: i.e., with strongly reliable time estimates, but not necessarily close to the optimum. Similarly, points close to the *y*-axis represent journeys which are fast but not necessarily robust: i.e., close to the optimum, but not necessarily with reliable time estimates. Points far away from the origin along the bisector of the first quadrant represent low quality solutions: the further away they are from the origin, and the lower their accuracy in estimates and the larger their distance from optimality.

As we see from Figure 7, the low quality of the journeys provided by RAPTOR(TT) is highlighted by a tendency to concentrate its solutions mostly along the bisector. On the other side, the improvement introduced by RcRAPTOR-AVG(GPS) over RAPTOR(TT) is witnessed by the fact that most of its journeys are roughly contained in a ball centered in the origin. In this framework, the main difference between 1-Fault-Aware RAPTOR(GPS) and RcRAPTOR-AVG(GPS) is that 1-Fault-Aware RAPTOR(GPS) tends to spread its solutions more towards the *x*-axis, thereby providing journeys which seem to be inherently more robust and less vulnerable to delays, at the cost of increasing their travel times (measured in terms of distance from optimality).

7. CONCLUSIONS

We studied how to improve a journey planning system for a metropolitan area with an available live GPS feed from all vehicles in the network. We evaluated the quality of the obtained journeys if we ignore the feed, and the amount we gain in quality by incorporating the feed into the journey planning system. We tested RAPTOR together with known and novel extensions. The key insight of our work is that we can improve the quality of the journeys computed, especially for short range queries. As for future work, we believe that one can also improve the results for long range queries by exploiting the GPS data more carefully. For this, however, it seems that we need more sophisticated prediction models for future delays.

8. REFERENCES

[1] L. Allulli, G. F. Italiano, and F. Santaroni. Exploiting GPS Data in Public Transport Journey Planners. In *SEA*, pp. 295–306. Springer, 2014.

[2] H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, and F. Viger. Fast Routing in Very Large Public Transportation Networks using Transfer Patterns. In *ESA*, pp. 290–301. Springer, 2010.

[3] H. Bast, D. Delling, A. V. Goldberg, M. Müller–Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route Planning in Transportation Networks. Technical Report MSR-TR-2014-4, Microsoft Research, 2014.

[4] H. Bast, J. Sternisko, and S. Storandt. Delay-Robustness of Transfer Patterns in Public Transportation Route Planning. In *ATMOS*, OASIcs, pp. 42–54, 2013.

[5] A. I. Bejan, R. J. Gibbens, D. Evans, A. R. Beresford, J. Bacon, and A. Friday. Statistical Modelling and Analysis of Sparse Bus Probe Data in Urban Areas. In *ITSC*, pp. 1256–1263. IEEE, 2010.

[6] A. Berger, A. Gebhardt, M. Müller–Hannemann, and M. Ostrowski. Stochastic Delay Prediction in Large Train Networks. In *ATMOS*, OASIcs, pp. 100–111, 2011.

[7] K. Böhmová, M. Mihalák, T. Pröger, R. Šrámek, and P. Widmayer. Robust Routing in Urban Public Transportation: How to Find Reliable Journeys Based on Past Observations. In *ATMOS*, OASIcs, pp. 27–41, 2013.

[8] D. Delling, B. Katz, and T. Pajor. Parallel Computation of Best Connections in Public Transportation Networks. *ACM Journal of Experimental Algorithmics*, 17(4):4.1–4.26, July 2012.

[9] D. Delling, T. Pajor, and R. F. Werneck. Round-Based Public Transit Routing. In *ALENEX*, pp. 130–140. SIAM, 2012.

[10] D. Delling, T. Pajor, and R. F. Werneck. Round-Based Public Transit Routing. *Transportation Science*, 2014. Accepted for publication.

[11] J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner. Intriguingly Simple and Fast Transit Routing. In *SEA*, pp. 43–54. Springer, 2013.

[12] Y. Disser, M. Müller–Hannemann, and M. Schnee. Multi-Criteria Shortest Paths in Time-Dependent Train Networks. In *WEA*, pp. 347–361. Springer, 2008.

[13] D. Firmani, G. F. Italiano, L. Laura, and F. Santaroni. Is Timetabling Routing Always Reliable for Public Transport? In *ATMOS*, OASIcs, pp. 15–26, 2013.

[14] M. Goerigk, S. Heße, M. Müller–Hannemann, and M. Schmidt. Recoverable Robust Timetable Information. In *ATMOS*, OASIcs, pp. 1–14, 2013.

[15] M. Goerigk, M. Knoth, M. Müller–Hannemann, M. Schmidt, and A. Schöbel. The Price of Strict and Light Robustness in Timetable Information. *Transportation Science*, 2013.

[16] T. Hunter, R. Herring, P. Abbeel, and A. Bayen. Path and travel time inference from GPS probe vehicle data. In *NIPS Workshop on Analyzing Networks and Learning with Graphs*, pp. 1–8, 2009.

[17] E. Jenelius and H. N. Koutsopoulos. Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological*, 53:64–81, 2013.

[18] M. H. Keyhani, M. Schnee, K. Weihe, and H.-P. Zorn. Reliability and Delay Distributions of Train Connections. In *ATMOS*, OASIcs, pp. 35–46, 2012.

[19] M. Müller–Hannemann and M. Schnee. Efficient Timetable Information in the Presence of Delays. In *Robust and Online Large-Scale Optimization*, pp. 249–272. Springer, 2009.

[20] B. Strasser and D. Wagner. Connection Scan Accelerated. In *ALENEX*, pp. 125–137. SIAM, 2014.

[21] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with Knowledge from the Physical World. In *SIGKDD*, pp. 316–324. ACM, 2011.