Vehicle Capacity-Aware Rerouting of Passengers in Delay Management

Matthias Müller-Hannemann
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg,
Von-Seckendorff-Platz 1, D 06120 Halle (Saale), Germany
muellerh@informatik.uni-halle.de

Ralf Rückert
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg,
Von-Seckendorff-Platz 1, D 06120 Halle (Saale), Germany
ralf.rueckert@informatik.uni-halle.de

Sebastian S. Schmidt
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg,
Von-Seckendorff-Platz 1, D 06120 Halle (Saale), Germany
sebastian.schmidt@informatik.uni-halle.de

Abstract

Due to the significant growth in passenger numbers, higher vehicle load factors and crowding become more and more of an issue in public transport. For safety reasons and because of an unsatisfactory discomfort, standing of passengers is rather limited in high-speed long-distance trains. In case of delays and (partially) cancelled trains, many passengers have to be rerouted. State-of-the-art rerouting merely focuses on minimizing delay at the destination of affected passengers but neglects limited vehicle capacities and crowding. Not considering capacities allows using highly efficient shortest path algorithms like RAPTOR or the connection scan algorithm (CSA).

In this paper, we study the more complicated scenario where passengers compete for scarce capacities. This can be modeled as a piece-wise linear, convex cost multi-source multi-commodity unsplittable flow problem where each passenger group which has to be rerouted corresponds to a commodity. We compare a path-based integer linear programming (ILP) model with a heuristic greedy approach. In experiments with instances from German long-distance train traffic, we quantify the importance of considering vehicle capacities in case of train cancellations. We observe a tradeoff: The ILP approach slightly outperforms the greedy approach and both are much better than capacity unaware rerouting in quality, while the greedy algorithm runs more than three times faster.

2012 ACM Subject Classification Applied computing → Transportation; Theory of computation → Discrete optimization; Mathematics of computing → Network flows

Keywords and phrases Delay management, passenger flows, vehicle capacities, rerouting

Digital Object Identifier 10.4230/OASIcs.ATMOS.2019.7

Funding This work has partially been supported by a DFG grant for the research group FOR 2083, project MU1482/7-2.

Acknowledgements The authors wish to thank Deutsche Bahn for providing test data.

1 Introduction

Recent years have shown significant growth in passenger numbers on public transport services in many countries. Due to political efforts to increase utilization of public transport in support of sustainability goals, further growth is to be expected. While congestion in metro systems of mega-cities during peak hours has been recognized as a challenge for many years, awareness of increased in-vehicle density as a problem also for the management of passenger flows in long-distance trains started only recently.

In this paper, we combine delay and disruption management with scarce vehicle capacities. In large and complex public transportation systems delays as well as disruptions occur frequently. Typical reasons are technical defects, construction work, bad weather conditions, exceptionally many passengers, accidents, and the like. As a consequence, passengers miss planned transfers which results in a significant delay at their destinations, in considerable dissatisfaction, and ultimately in economic loss for the railway company. In delay management, train dispatchers have to decide which trains shall wait for delayed incoming trains in order to maintain connections for passengers. Such problems are challenging for several reasons: One has to deal with large-scale networks subject to dynamically changing, partially incomplete and imprecise information about current delays and their propagation, and solutions are required in almost real-time. In an on-going joint research project with Deutsche Bahn, we are working on the development of a decision support system for dispatchers which shall help to find optimal waiting decisions from a passenger's point of view. A key assumption is that detailed information about passenger flows is available, that is, for each passenger the planned route is known. Such passenger flows can be based on sold tickets or statistically validated demand models. In our long-term project, we have built a prototype for an optimized passenger-friendly disposition system, named PANDA [16, 19]. The acronym PANDA abbreviates Passenger-Aware Novel Dispatching Assistance. It is designed to provide train dispatchers with detailed real-time information about the current passenger flow and the multi-dimensional impact of waiting decisions in case of train delays and cancellations. If trains are cancelled or connections cannot be maintained, passengers have to be rerouted. State-of-the-art solutions determine new routes for passengers that are optimized subject to earliest arrival at the planned destination with few transfers as a secondary criterion. Technically, this requires the efficient solution of large-scale multi-criteria shortest path problems in suitably designed event-activity networks. Recent progress in shortest path algorithms for such applications allows to solve such problems in a few milliseconds per instance, for example by using RAPTOR [5] or the connection scan algorithm (CSA) [6, 7]. Capacity constraints, however, are widely neglected in previous work. Considering the available free capacity to avoid overcrowded trains leads to several, more challenging combinatorial optimization problems. With respect to capacities, we may distinguish between hard and soft capacities. For each vehicle, there is a designated number of available seats. This gives a soft capacity beyond which it becomes more and more uncomfortable to travel. At a certain threshold, the hard capacity, a vehicle becomes so crowded that it is not allowed to run anymore for security reasons.[1] Key drivers for crowding discomfort of passengers are dissatisfaction with standing and not being seated, fewer opportunities to make use of the time during the journey, and the physical closeness of other travellers per se [13].

Goals and contribution. The main use case and focus of this work are cancelled or partially cancelled trains where many (up to several hundred) passengers have to be rerouted simultaneously. A second use case are passengers with missed connections due to wait-depart decisions in delay management. For the most important train connections, also large numbers of passengers are affected by a single decision.

A crucial issue concerns the model how passengers behave. If we assume that passengers behave selfishly and inform themselves individually and independently about alternative routes, we have only limited capabilities to avoid overcrowded trains. All we can do in such a

[1] According to Richard Lutz, Chief Executive Officer (CEO) of Deutsche Bahn, overfull long-distance trains of Deutsche Bahn have to be stopped and evacuated half a dozen times a week (Handelsblatt of April 19, 2018).

scenario is to only recommend trains which have enough free capacity and to put preference on connections with lower seat occupation. To support such a goal, one can still solve shortest path problems. In an attempt to avoid violations of hard capacities, we can simply forbid all arcs in the event-activity model which would lead to overfull vehicles. Moreover, the objective function can be modified so that it prefers trains with larger free capacity.

Different optimization problems arise if we take the perspective of a recommending system which tries to achieve a system optimum, that is, a solution which minimizes the overall inconvenience for all affected passengers. Inconvenience can be expressed in several ways, the simplest version being the total delay at the destination, summed up over all passengers. Overcrowding of trains can be penalized with the help of convex cost functions. For such scenarios, we have to solve some large-scale integral minimum cost multi-commodity flow problems, where each group of passengers sharing the same origin and destination corresponds to some commodity. Since passengers groups want to travel together, we have to consider versions of unsplittable flow problems. Such problems are well-known to be NP-hard optimization problems. The most common approach to solve them is integer linear programming (ILP) with a path-based formulation and column generation. While the underlying network is fairly large, the number of commodities to be considered is typically of moderate size. Moreover, for each commodity, there is only a limited number of "reasonable" alternative paths to which one can heuristically restrict the search. We follow this general approach, and in addition, we will also consider a fast greedy rerouting scheme. With this work we want to tackle the following research questions:

1. How relevant is crowding-aware rerouting in long-distance trains already today? How much more important will it be with rising numbers of passengers?
2. Comparing a capacity-aware greedy rerouting with a minimum cost multi-commodity flow model, how much do we lose in quality if we use a greedy algorithm?
3. Can we solve the instances of unsplittable flow problems fast enough in practice?

Due to our cooperation with DB Fernverkehr, we concentrate on long-distance trains. Our main results are as follows. First, we observe that ignoring vehicle capacities would guide many passengers into overfull trains. This effect will become more severe with rising passenger numbers. Conversely, with our models we can reduce passenger inconvenience to a large amount. Second, the ILP solution is slightly better in quality than the greedy approach, but the greedy approach is about three times as fast. Third, while the ILP problems can be solved very easily within milliseconds, the required computation of alternative paths turns out to be the bottleneck. Severe cases of train cancellations with several hundred passenger groups require on average less than 85 seconds of computation time for our ILP approach.

Related work. Delay management has been studied very intensively in the literature, see the recent survey [10]. Based on event-activity networks most of these approaches model delay management by integer linear programming (ILP) and consider offline versions of the problem, where all delays are known before the optimization process starts [21, 22]. First approaches considered simplified versions and assumed that passengers who miss a connection wait for the next connection on the same line. Integrated passenger rerouting has been considered by [9, 10, 20]. The integration of passenger rerouting into an ILP formulation leads to a considerable blow-up, making these formulations very large. With today's techniques for integer programming, the handling of several hundred thousands of passenger routes seems to be impossible in an online setting. Dollevoet and Huisman [8] propose several fast heuristics and introduce an iterative ILP approach which comes close to the exact solution but is significantly faster. However, it remains open whether their iterative ILP approach

scales well to large-scale networks. All previous delay management models have in common that they do not consider vehicle capacities and crowding. Models and algorithms for efficient passenger routing in public transport (timetable information) have intensively been discussed in the literature, see for example the surveys by Müller-Hannemann et al. [17] and the more recent one on efficient shortest path algorithms by Bast et al. [4]. Among the many recent approaches for shortest path computation in public transportation networks, most relevant for this work are those which are well suited to a dynamic online scenario. As they require no heavy preprocessing, the above-mentioned approaches RAPTOR [5] and an extension of CSA [6, 7] serve very well for minimizing earliest arrival time and the number of transfers. RAPTOR can be extended to determine the Pareto set of optimal solutions for additional criteria, for example reliability and ticket price (McRaptor). Discomfort of crowding as an additional criterion has to the best of our knowledge not been considered in a multi-criteria setting. There has been, however, related work, in load balancing of passenger flows. For example, Huang et al. [14] study route guidance for passengers as a means to reduce in-vehicle congestion. The problem of finding alternative routes has found quite some attention before. One possibility is to search for the top k shortest paths. Since event-activity networks are directed acyclic graphs, these paths can be found and output by Eppstein's algorithm in $O(n \log n + m + kn)$ time in networks with n vertices and m arcs [11]. As the top k shortest paths may be too similar, several attempts have been done to find sets of paths with limited overlap. One such approach is first to compute a large set of candidate paths and then to filter candidates with respect to some similarity measure, for example [1].

Multi-commodity flows and unsplittable flow problems have been extensively studied in the literature [2, 23]. In general, they are strongly NP-hard since many combinatorial problems, including disjoint paths, can be reduced to it. Classical applications of unsplittable flow problems include, for example, bandwidth packing problems in telecommunication networks or express package delivery problems in logistics [3]. For solving large-scale instances of unsplittable flow problems, path-based formulations have advantages over arc-based formulations [3]. Since the number of paths grows exponentially with the size of the graph, exact solutions usually require column generation. Barnhart et al. [3] provide seminal work on column generation and branch-and-price-and-cut algorithms for unsplittable flows. Fortz et al. [12] discuss models for piecewise linear cost versions of the unsplittable multi-commodity flow problem. Wang [23] provides a recent survey on solution methods for multi-commodity network flow problems.

Overview. The remainder of this paper is organized as follows. In Section 2, we present our multi-commodity flow model. We start with general considerations and model assumptions. Afterwards, we develop and explain step-by-step the underlying event-activity network, our modelling of capacities and cost functions, and the resulting integer programming formulation. Then, we present a fast greedy heuristic and, finally, we describe our approach for the computation of candidates for alternative paths. Our computational study with many instances from Deutsche Bahn is reported in Section 3. Finally, we summarize and conclude with future work.

2 Multi-Commodity Flow Model

In this section, we develop our approach for the simultaneous rerouting of passengers with respect to limited vehicle capacities.

2.1 Basic considerations and assumptions

Our model is based on the following considerations and assumptions:

- In case of a disruption, our task is to find valid alternative routes for *all* affected passengers whose planned connection becomes invalid. If we cannot find an "acceptable" alternative (say, with at most two hours of delay at the destination), this imposes a high cost for compensation.
- Only directly affected passengers are rerouted. We assume that all others who are not forced to change plans keep their planned route. In some cases, delayed trains may offer new opportunities for passengers to optimize their routes, but this issue is ignored in our model.
- Groups of passengers have planned to travel together (for instance, couples, families, school classes). They certainly have to stay together also in their new route. This implies that we have to consider unsplittable flow models. Moreover, many passengers share the same origin and destination even if they are personally unrelated. They may be treated as a group since recommending different alternatives to them might be hard to explain and communicate without personalized route guidance.
- Rerouting of passengers comes with several disadvantages. Passengers may lose their seat reservation and have to enter more crowded trains. The valuation of discomfort is very subjective and varies widely between passengers. It depends individually on personal circumstances (like age and healthiness), the reason for traveling, and several other factors [15]. Nevertheless, to keep the model simple enough, we use the same general utility functions for all passengers.
- For simplicity, we do not make a distinction between first and second class travelers.
- Train tickets are often bound to a specific connection for which they are booked. In case of disruption, we assume that passengers may choose any train and any connection (no restrictions due to ticket regulations apply).
- We restrict the set of eligible alternative paths to "reasonable" ones: i.e. we consider only paths where passengers arrive at their final destination at most 120 minutes after the earliest feasible arrival time. We also exclude paths with too many train changes, and the upper limit is at most six transfers.
- Train cancellations or missed transfers only become known at short notice, at a certain event-specific release time. Since passengers can react only after they become aware of a need to find an alternative route, we require that replanning can alter the original route only after this release time. Several cases are possible:
  1. The passenger has not yet started traveling. In this case, we assume that the passenger arrives at the station where the original route would have started more or less just in time for the planned train. Thus, an alternative route may not start earlier and should begin at the same station (although an initial footpath is allowed).
  2. The passenger is already traveling. Then, based on the current location (in some specific train or at a station) and time, an alternative connection must be compatible with the initial part of the original route.

Event-activity network

The given train schedule and the set of passenger routes can be modeled with the help of a so-called *event-activity network (EAN)* $N = (V, A)$, a directed acyclic graph with vertex set V and arc set A. The vertices of the network correspond to the set of all arrival and departure events of the given schedule. Arcs of the network model order relations between events. We distinguish between different types of arcs ("activities"):

- *driving arcs*, modeling the driving of a specific train from one stop to its very next stop,
- *dwelling arcs*, modeling a train standing at a platform and allowing passengers to deboard or board the train, and
- *transfer arcs*, modeling the possibility for passengers to switch between two trains at the same or nearby stations.

Passenger routes correspond to paths in N from a departure event to some arrival event. Let K be the set of passenger groups which have to be rerouted due to a train cancellation or broken transfer. For $k \in K$, denote by d_k the size of group k. Denote by t_k the intended destination, i.e. the final station of the planned route. Likewise, denote by s_k the origin ("source") of this group with respect to the time of replanning. The origin is the first station of the planned route if the journey has not yet started. Otherwise, it denotes the station where the group is currently waiting or the very next station at which they will arrive with their current train.

We extend the event-activity network N by adding a source and a target "event" for each passenger group $k \in K$. Each source s_k is connected to all departure events at the same station which can be reached by the group. If footpaths to nearby stations exist, we also connect s_k to the reachable departure events at these stations. At the target station, we connect all arrival events with t_k. In summary, we seek for each group a path which starts at source s_k and and ends at target t_k. For instances with very high load and in particular for large groups of passengers, no feasible capacity-respecting path may exist. To make sure, that every instance has a feasible solution, we add for each pair (s_k, t_k) some direct "no-route" arc of infinite capacity but very high costs, so that such arcs are only chosen if no other path is available.

Capacities and cost functions

With every arc a which corresponds to a driving or dwelling activity of a train, we can associate a nominal seat capacity $cap(a)$. Recall that we do not distinguish between first and second class seats for simplicity. The hard capacity for such an arc is set as $\beta \cdot cap(a)$ where $\beta \geq 1$ is a parameter specifying the maximal overload acceptable for security reasons. For high-speed trains, choosing $\beta = 1.2$ may be a reasonable choice (and is used in our experiments).

For rerouting, we have to consider the *free capacity* which remains if we subtract the number of those passengers which are not affected by rerouting. Thus, if $load(a)$ denotes the current number of passengers on arc a, we obtain an upper bound $u_a = \max\{0, \beta \cdot cap(a) - load(a)\}$. (Arcs of overloaded segments with $u_a = 0$ are excluded from the model.) All arcs in A not corresponding to driving or dwelling activities of trains have unlimited capacity.

For arc $a \in A$, let $C_a(x_a)$ be a piece-wise linear convex cost function. For simplicity in notation, we assume that each cost function has exactly b linear segments, but we allow empty segments to model cost functions with fewer breakpoints. With $0 = u_a^0 \leq u_a^1 \leq u_a^2 \leq \cdots \leq u_a^b = u_a$ we denote the breakpoints of the function. The cost varies linearly in the interval $[u_a^{i-1}, u_a^i]$ with slope c_a^i. Slopes are strictly increasing, i.e. $c_a^1 < c_a^2 < \cdots < c_a^b$ (unless they are $+\infty$).

As a cost function, we use a kind of generalized or *perceived travel time* which penalizes transfers and crowding discomfort. For the latter, we use the time-multiplier method [18]. The cost function's basic unit is travel time in minutes. Traveling, dwelling and transfer arcs each have a certain duration $dur(a)$. Train changes are penalized with α extra minutes per transfer. Traveling in the train (i.e. being on traveling or dwelling arcs) is penalized with respect to the load. For the segment of the piecewise linear cost function, we have penalty factors $\gamma_1 < \gamma_2 < \cdots < \gamma_b$. Cost parameters are set as follows.

- For traveling and dwelling arcs we set:

$$c_a^1 = (1+\gamma_1)\cdot dur(a), c_a^2 = (1+\gamma_2)\cdot dur(a), \ldots, c_a^b = (1+\gamma_b)\cdot dur(a).$$

- Transfer arcs are uncapacitated and have only a single finite cost segment, thus we set

$$c_a^1 = dur(a) + \alpha, c_a^2 = +\infty, \ldots, c_a^b = +\infty.$$

- "No-route arcs" have very high costs, say $c_a^1 = 10000, c_a^2 = +\infty, \ldots, c_a^b = +\infty$.
- All remaining arcs have zero costs.

2.2 Integer linear programming formulation

We are now ready to formulate the unsplittable flow problem as an integer linear program. Denote by $P(k)$ be the set of all paths from s_k to t_k from which the group has to select exactly one. We use binary decision variables y_p^k where $y_p^k = 1$ if group k selects path $p \in P(k)$, and $y_p^k = 0$ otherwise. Let δ_a^p be an arc-path indicator variable that equals 1 if arc a is contained in path p. For arc $a \in A$, let x_a^k be the size of the flow on arc a of commodity k, and $x_a = \sum_{k\in K} x_a^k$ be the total flow on this arc.

A classical transformation of piecewise convex flows to standard flow with linear costs is to replace each arc a by a set of b parallel arcs [2]. The idea is to decompose the flow x_a into flows on the segments between neighboring breakpoints of the cost function. Define

$$f_a^i = \begin{cases} 0 & \text{if } x_a \le u_a^{i-1} \\ x_a - u_a^{i-1} & \text{if } u_a^{i-1} < x_a \le u_a^i \\ u_a^i - u_a^{i-1} & \text{if } x_a \ge u_a^i. \end{cases}$$

This implies $x_a = \sum_{i=1}^b f_a^i$ and $C_a(x_a) = \sum_{i=1}^b c_a^i f_a^i$. The path flow formulation is then:

$$\min \sum_{a\in A}\sum_{i=1}^{b} c_a^i f_a^i \tag{1}$$

subject to

$$\sum_{p\in P(k)} y_p^k = 1 \qquad \text{for all } k \in K \tag{2}$$

$$\sum_{k\in K}\sum_{p\in P(k)} d_k y_p^k \delta_a^p = \sum_{i=1}^{b} f_a^i \qquad \text{for all } a \in A \tag{3}$$

$$f_a^i \le u_a^i \quad \text{for all } a \in A \text{ and all } i = 1,2,\ldots,b \tag{4}$$

$$f_a^i \ge 0 \quad \text{for all } a \in A \text{ and all } i = 1,2,\ldots,b \tag{5}$$

$$y_p^k \in \{0,1\} \quad \text{for all } k \in K \text{ and all } p \in P(k) \tag{6}$$

Equations (2) ensure that exactly one path has to be chosen for each commodity. Equations (3) express that the total flow $x_a = \sum_{i=1}^b f_a^i$ on arc a equals the sum of chosen paths using this arc, weighted by the demands d_k of the commodities. Capacity constraints of all flow segments are given by Inequalities (4), while non-negativity of all flow variables is provided by Inequalities (5). The integrality of all flow variables f_a^i is implied by the integrality of the left-hand-side of Equations (3) and the strict increase in slope values c_a^i.

2.3 A fast greedy heuristic

The exact solution of the unsplittable flow problem is NP-hard, although the instances arising in our applications seem to be quite well solvable by state-of-the-art ILP solvers (see our experiments below). However, it requires the computation of many alternative paths in a first step which is computationally expensive.

Therefore, we suggest a simple, but fast greedy heuristic: process the passenger groups one after another (in some random order). For each group $k \in K$, compute the shortest alternative path with respect to the perceived travel time, subject to current free capacities. Assign the passengers of this group to the new route, update the capacities, and continue with the next group.

2.4 Path set computation

The set of all s-t–paths can be exponentially large. However, only a small number of them is sufficiently attractive for passengers so that they are actually used. Therefore, instead of working with the full set of paths $P(k)$ for each commodity, we heuristically restrict the search to a carefully selected small set of paths.

To this end, we first compute the Pareto set of shortest paths with respect to three criteria travel time, number of transfers, and some measure of inconvenience. We propose two variants:

1. PARETO1: we use the same perceived travel time function as inconvenience measure as in the greedy approach.
2. PARETO2: we consider a measure which focuses on load. The load of an arc with capacity restrictions (a *capacitated arc*) is defined with the same cost parameters as in the ILP. The load of a path is then defined as the sum of loads on its capacitated arcs.

In PARETO1, the first and third criteria are highly correlated, resulting in relatively small path sets. In contrast, the travel time and load are much less correlated (although the load costs of an arc also depend on its duration), leading to slightly larger Pareto sets. To increase the likelihood of finding feasible paths, we finally add the path set of the greedy approach. As mentioned above, the Pareto sets are pruned in both variants such that only paths with at most six transfers and at most two hours of extra delay are maintained. All other paths are considered as unacceptable.

3 Experiments

In this section, we report our findings with experiments on many large-scale test instances.

3.1 Test instances, implementation details, and experimental setup

Test instances

Our experiments are based on the timetable of Germany in 2019. For each day, passenger flow data for travelers using long-distance trains are provided by Deutsche Bahn. They are based on sold tickets until the day before and an estimation of short-term ticket buyers. Our capacity data is restricted to that of long-distance trains, for other trains we assume infinite capacity. We selected five test days in April 2019, in the week from Monday 22 to Friday 26. For the runtime measurements we only used the Friday. The average number of passengers in our passenger flow data is about 410,000 passengers (minimum 280,000 and maximum 540,000). In order to create meaningful and relatively hard test instances, we

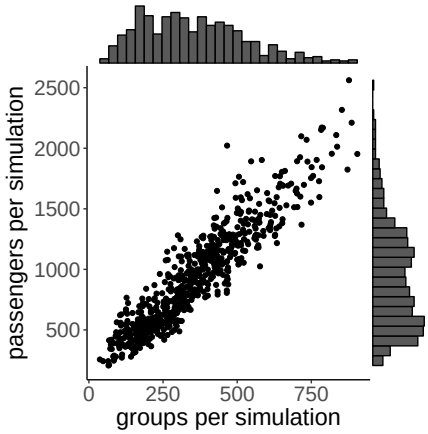

Figure 1 Number of groups and passengers per instance.

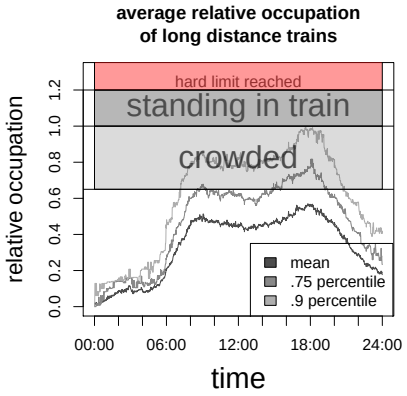

Figure 2 Vehicle load over time of long-distance trains in Germany.

concentrate on train cancellations with many affected passengers and groups which have to be rerouted. To this end, we randomly selected a subset of trains with the property that their passenger load reaches at least 65% of its capacity. Each train cancellation is studied in isolation as an independent test instance. Overall, we have a test set composed of 653 train cancellations. Figure 1 shows the distribution of the number of affected passengers and groups per instance. The average number of affected groups is 351.4, the average number of affected passengers is 922.84.

Specific cost functions and parameters

The following parameter settings are used in all experiments. In our basic model, we use only three different segments for capacitated arcs. Recall that the hard limit u_a is chosen with respect to $\beta = 1.2$, i.e. 120% of the nominal vehicle capacity. The interval from 0 to the hard upper limit u_a is divided as follows:

1. The train is currently occupied by at most 65% of its capacity. We set $u_a^1 = 65/120 \cdot u_a$. This is considered as a non-crowding scenario and imposes no crowding penalties, i.e. $\gamma_1 = 0$.
2. The current load is between 65% and 100% capacity. Every passenger finds a seat, but with limited choice. We set $u_a^2 = 100/120 \cdot u_a$. Here we impose a crowding penalty of $\gamma_2 = .2$, i.e. we impose .2 extra minutes per minute of travel time.
3. The available capacity is exceeded, some passengers have to stand. The penalty for standing is $\gamma_3 = 1$, i.e. one extra minute per minute of travel time.

Experimental setup

Our code has been written in C++, it is compiled with gcc 8.3 and run under Arch Linux x86_64 with packages from Mai 2019. All runs are executed on a four core plus hyper-threading Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz machine. Shortest path problems in the event-activity network are computed by an implementation of RAPTOR. ILPs are solved with Gurobi Optimizer 8.1[2].

[2] `http://www.gurobi.com`

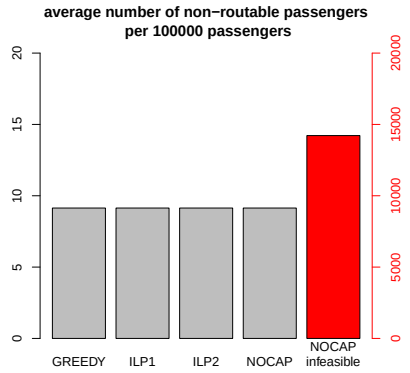

Figure 3 The number of passengers ending up without a valid route. For NOCAP, we have to distinguish passengers with no route, and others with a route which violates hard capacities (last column, right scale).

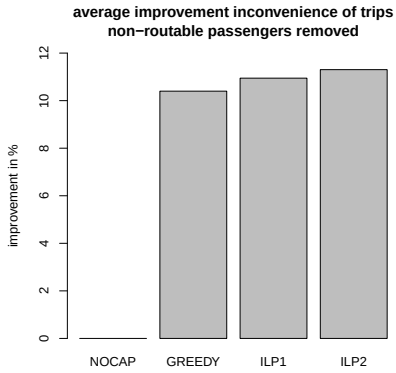

Figure 4 The mean improvement in inconvenience cost in %.

We compare the following four approaches:

1. **NOCAP**: this refers to the approach of capacity-unaware rerouting. Passengers are simply rerouted to a path with earliest arrival time at their destination. Capacity constraints may be violated.
2. **GREEDY**: passenger groups are rerouted greedily as described in Section 2.3.
3. **ILP1**: this refers to the ILP model where the path set is chosen as PARETO1.
4. **ILP2**: the same ILP model is used but the path set is chosen as PARETO2.

3.2 Experimental Results

Question 1: How relevant is the consideration of scarce vehicle capacities in rerouting?

Considering the initial load (before rerouting) of long-distance trains, the average vehicle load of 44.71% (within the interval from 6am to 10pm) may falsely suggest that there is probably no severe problem with scarce capacities. However, if we look more carefully into the distribution we observe that the average load changes considerably during the course of the day, see Figure 2. Many of the most crowded trains run in the afternoon hours and close to their maximum seat capacity.

The problem with scarce capacities becomes apparent when we evaluate the traditional capacity-unaware rerouting scheme. Figure 3 shows the number of passengers for which we cannot find a (valid) route. For capacity-unaware rerouting (NOCAP), we have a few cases without any route (column 4), and many more cases, about 15% of all passengers, with an alternative route violating hard train capacities (column 5). In other words, that many affected passengers will by routed into some overfull train whenever capacities are ignored!

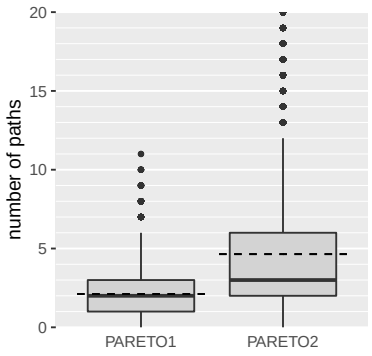

Figure 5 Comparison of PARETO1 and PARETO2: Distribution of sizes of path sets.

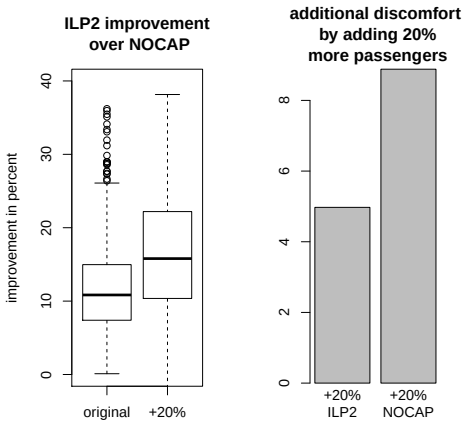

Figure 6 Comparison of original passenger flows and flows scaled by +20%. Left: The benefit of ILP2 over GREEDY. Right: comparison of additional discomfort ILP2 and NOCAP.

Question 2: How large is the improvement for passengers if we apply capacity-aware routing?

With the capacity-respecting variants, we can strongly reduce the number of passengers without a valid route, see again Figure 3. For less than 0.01% of the passengers no valid route exists (for all four algorithmic variants). Such cases occur for some rare connections without a feasible alternative within the next 24 hours.

We also evaluated the improvement in inconvenience costs over the NOCAP baseline version. Concentrating only on the subset of cases where all variants find valid routes for all passenger groups, we see that GREEDY reduces the total inconvenience costs by more than 10.5% in comparison with NOCAP. Even better mean improvements of about 11% and 11.5% over the baseline NOCAP are obtained with ILP1 and ILP2, respectively, see Figure 4.

The better average quality of ILP2 over ILP1 can be partially explained by the underlying path sets. As Figure 5 shows, the mean size of the PARETO1 path set is smaller than PARETO2, thus leading to fewer rerouting options within ILP1 in comparison with ILP2. Even better solutions can be expected if we further increase the path sets.

Question 3: What happens if passenger numbers increase by 20%?

To answer this question, we take the original passenger flows and scale them up by 20%. Figure 6 shows that the ILP2 solution is continuing to outperform the NOCAP solution on average by 15.0%. The change in total discomfort by passengers, however, is significant. Adding 20% more passengers increases the discomfort experienced in the NOCAP strategy by 9% while the ILP2 model does only produce 5% more discomfort for the same passengers. Thus, we conclude that capacity-aware routing will become more important with rising passenger numbers.

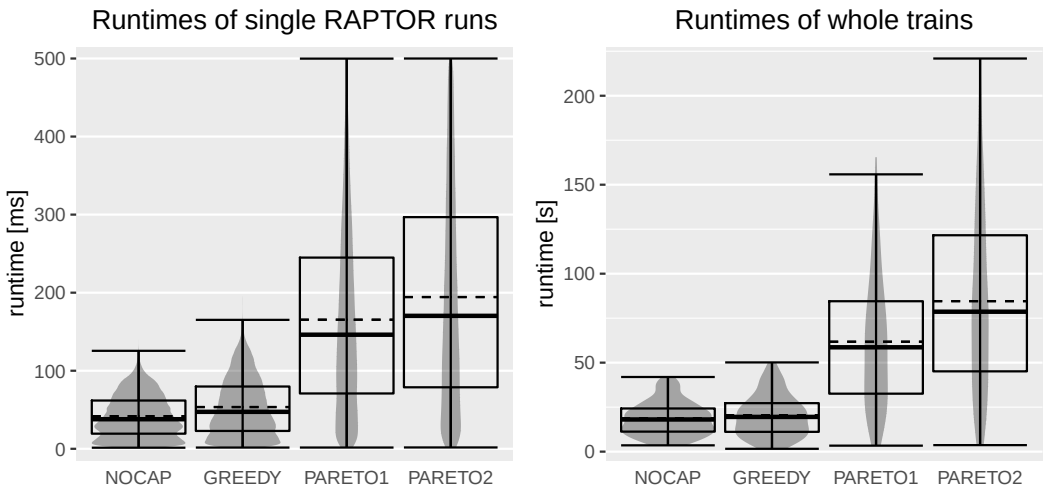

Figure 7 Runtime distributions of single runs of RAPTOR (left) and for whole trains (right) for the four different variants NOCAP, GREEDY, PARETO1, and PARETO2.

Question 4: How efficient are the proposed approaches?

Let us start with some good news: When the underlying path sets are generated, it turns out that solving the ILP resulting instances is very easy for state-of-the-art solver gurobi, since all of them can be solved in few milliseconds.

The expensive part, however, is the computation of candidate paths for all passenger groups. Figure 7 shows runtime distributions as violin plots for single runs of RAPTOR for the four different variants NOCAP, GREEDY, PARETO1 and PARETO2. The mean runtimes are 42ms for NOCAP, 53ms for GREEDY, 166ms for PARETO1, and 228ms for PARETO2. The mean runtime to compute the path sets for all affected groups of a train cancellation is 19s for NOCAP, 20s for GREEDY, 62s for PARETO1 and 85s for PARETO2. We consider such runtimes as feasible for practical use. Improvements of running times are possible by further fine-tuning our implementations.

4 Conclusions and Future Work

In this paper we study the impact of limited vehicle capacities on the rerouting of passengers in case of train cancellations. We propose a convex cost unsplittable flow formulation. First experiments with restricted path sets already show significant improvements over previous, capacity unaware approaches.

In the present work, we solve the unsplittable flow problems with respect to a carefully selected fixed choice of paths. This could be extended by column generation (which amounts to solving a single-criterion shortest path problem for each commodity). The computational bottleneck is the efficient computation of path sets. With respect to our implementation there is certainly room for further improvement for the multi-criteria versions of RAPTOR. While the set of greedy paths has to be computed sequentially, the path set for passenger groups in the multi-criteria setting are independent and can be easily parallized.

We plan to extend our work in several ways. A first natural extension is to study different convex cost functions. Since our focus has been on the most extreme cases (up to several hundred affected passenger groups), a second extension concerns evaluations of real train

cancellations and wait-depart decisions, and similar use cases. Third, we are interested in the price of restricting to unsplittable flows. By how much can we improve solutions if we allow splitting of groups? We could either consider the linear programming relaxation of our ILP models as a lower bound or a closely related classical multi-commodity flow formulation.

References

1 Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Alternative Routes in Road Networks. *J. Experimental Algorithmics*, 18:1.3:1.1–1.3:1.17, 2013. doi:10.1145/2444016.2444019.

2 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows*. Prentice Hall, Inc., 1993.

3 Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. *Oper. Res.*, 48(2):318–326, 2000. doi:10.1287/opre.48.2.318.12378.

4 Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route Planning in Transportation Networks. In Lasse Kliemann and Peter Sanders, editors, *Algorithm Engineering - Selected Results and Surveys*, volume 9220 of *Lecture Notes in Computer Science*, pages 19–80. Springer, 2016. doi:10.1007/978-3-319-49487-6_2.

5 Daniel Delling, Thomas Pajor, and Renato F. Werneck. Round-Based Public Transit Routing. *Transportation Science*, 49(3):591–604, 2015. doi:10.1287/trsc.2014.0534.

6 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Intriguingly Simple and Fast Transit Routing. In Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela, editors, *Experimental Algorithms, SEA 2013*, volume 7933 of *Lecture Notes in Computer Science*, pages 43–54. Springer, 2013. doi:10.1007/978-3-642-38527-8_6.

7 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Connection Scan Algorithm. *ACM Journal of Experimental Algorithmics*, 23:1.7:1–1.7:56, 2018. doi:10.1145/3274661.

8 Twan Dollevoet and Dennis Huisman. Fast heuristics for delay management with passenger rerouting. *Public Transport*, 6(1-2):67–84, 2014. doi:10.1007/s12469-013-0076-6.

9 Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay Management with Re-Routing of Passengers. *Transportation Science*, 46(1):74–89, 2012. doi:10.1287/trsc.1110.0375.

10 Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay propagation and delay management in transportation networks. In Ralf Borndörfer, Torsten Klug, Leonardo Lamorgese, Carlo Mannino, Markus Reuther, and Thomas Schlechte, editors, *Handbook of Optimization in the Railway Industry*, pages 285–317. Springer International Publishing, 2018.

11 David Eppstein. Finding the K Shortest Paths. *SIAM J. Comput.*, 28(2):652–673, 1999. doi:10.1137/S0097539795290477.

12 Bernard Fortz, Luís Gouveia, and Martim Joyce-Moniz. Models for the piecewise linear unsplittable multicommodity flow problems. *European Journal of Operational Research*, 261(1):30–42, 2017. doi:10.1016/j.ejor.2017.01.051.

13 Luke Haywood, Martin Koning, and Guillaume Monchambert. Crowding in public transport: Who cares and why? *Transportation Research Part A: Policy and Practice*, 100:215–227, 2017.

14 Zhiyuan Huang, Ruihua Xu, Wei D. Fan, Feng Zhou, and Wei Liu. Service-Oriented Load Balancing Approach to Alleviating Peak-Hour Congestion in a Metro Network Based on Multi-Path Accessibility. *Sustainability*, 11:1293, 2019. doi:10.3390/su11051293.

15 Zheng Li and David A. Hensher. Crowding in Public Transport: A review of objective and subjective measures. *Journal of Public Transportation*, 16:107–134, 2013.

16 Matthias Müller-Hannemann and Ralf Rückert. Dynamic Event-Activity Networks in Public Transportation — Timetable Information and Delay Management. *Datenbank-Spektrum*, 17:131–137, 2017. doi:10.1007/s13222-017-0252-y.

17 Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. Timetable Information: Models and Algorithms. In *Algorithmic Methods for Railway Optimization*, volume 4395 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2007.

18 Adam J. Pel, Nick H. Bel, and Marits Pieters. Including passengers' response to crowding in the Dutch national train passenger assignment model. *Transportation Research Part A: Policy and Practice*, 66:111–126, 2014. doi:10.1016/j.tra.2014.05.007.

19 Ralf Rückert, Martin Lemnian, Christoph Blendinger, Steffen Rechner, and Matthias Müller-Hannemann. PANDA: a software tool for improved train dispatching with focus on passenger flows. *Public Transport*, 9(1):307–324, 2017. doi:10.1007/s12469-016-0140-0.

20 Marie Schmidt. Simultaneous optimization of delay management decisions and passenger routes. *Public Transport*, 5:125–147, 2013. doi:10.1007/s12469-013-0069-5.

21 Anita Schöbel. A Model for the Delay Management Problem based on Mixed-Integer Programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

22 Anita Schöbel. *Customer-oriented optimization in public transportation*. Springer, Berlin, 2006.

23 I-Lin Wang. Multicommodity Network Flows: A Survey, Part II: Solution Methods. *International Journal of Operations Research*, 15:155–173, 2018.