Sensitivity Analysis and Coupled Decisions in Passenger Flow-Based Train Dispatching*

Martin Lemnian[1], Matthias Müller-Hannemann[2], and Ralf Rückert[3]

1 **Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg, Von-Seckendorff-Platz 1, 06120 Halle, Germany,**
mlemnian@informatik.uni-halle.de

2 **Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg, Von-Seckendorff-Platz 1, 06120 Halle, Germany,**
muellerh@informatik.uni-halle.de

3 **Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg, Von-Seckendorff-Platz 1, 06120 Halle, Germany,**
rueckert@informatik.uni-halle.de

Abstract

Frequent train delays make passenger-oriented train dispatching a task of high practical relevance. In case of delays, dispatchers have to decide whether trains should wait for one or several delayed feeder trains or should depart on time. To support dispatchers, we have recently introduced the train dispatching framework PANDA (CASPT 2015).

In this paper, we present and evaluate two enhancements which are also of general interest. First, we study the sensitivity of waiting decisions with respect to the accuracy of passenger flow data. More specifically, we develop an integer linear programming formulation for the following optimization problem: Given a critical transfer, what is the minimum number of passengers we have to add or to subtract from the given passenger flow such that the decision would change from waiting to non-waiting or vice versa? Based on experiments with realistic passenger flows and delay data from 2015 in Germany, an important empirical finding is that a significant fraction of all decisions is highly sensitive to small changes in passenger flow composition. Hence, very accurate passenger flows are needed in these cases.

Second, we investigate the practical value of more sophisticated simulations. A simple strategy evaluates the effect of a waiting decision of some critical transfer on passenger delay subject to the assumption that all subsequent decisions are taken according to standard waiting time rules, as usually employed by railway companies like Deutsche Bahn. Here we analyze the impact of a higher level of simulation where waiting decisions for a critical transfer are considered jointly with one or more other decisions for subsequent transfers. We learn that such "coupled decisions" lead to improved solution in about 6.3% of all considered cases.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems; G.2.2 Graph Theory (Graph algorithms; Network problems)

Keywords and phrases train delays, event-activity model, multi-criteria decisions, passenger flows, sensitivity analysis

Digital Object Identifier 10.4230/OASIcs.ATMOS.2016.2

* This work has been partially supported by DFG grant MU 1482/7-1 and Deutsche Bahn. We thank Deutsche Bahn for providing us with test data.

16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'16).
Editors: Marc Goerigk and Renato Werneck; Article No. 2; pp. 2:1–2:15

Open Access Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Public railway passenger transport is a key for greater mobility. Every day millions of passengers choose to travel by train and rely on the quality of service offered by railway companies. In daily operations, the quality of service is hindered by the fact that train delays and disruptions occur frequently. For passengers this often means that they miss some transfer and arrive delayed at their destination. To improve and to maintain their quality of service, railway companies employ train dispatchers who monitor delays and manually decide which trains shall wait for delayed incoming trains in order to maintain connections for passengers. Since waiting decisions induce further delays which propagate through the network, train delay management becomes a challenging and highly complex optimization problem. In this paper, we focus on coping with small and medium-size delays, while disruption management deals with more severe cases.

Recently, we introduced the decision support tool PANDA (Passenger Aware Novel Dispatching Assistance) [11] which has been developed together with Deutsche Bahn. The purpose of this tool is to provide dispatchers with information about which transfers are critical and require their attention, and the impact of waiting decisions gained from simulation. The evaluation of estimated arrival delays at the final destinations of all affected passengers is the basis for a qualified recommendation to wait or not to wait, and where applicable, how many minutes to wait. The PANDA prototype has been successfully tested in two field studies. In this paper, we introduce and study two enhancements of PANDA. As additional features we provide a sensitivity analysis and a more sophisticated simulation.

In practice, train disposition always has to deal with fuzzy and uncertain data. The current delay scenario and predictions about arrival and departure times change from minute to minute. Thus, we have to work with incomplete and estimated data. Since real-time passenger flow data will only be available in the future, our passenger flows are based on resource planning data, estimated by experts of Deutsche Bahn. This data is likely to be quite accurate, but for a specific day of operations we have to expect deviations. For example, it might be that a whole group of people, say a school class, is sitting in a train while we assume that there is none. Therefore, we want to investigate how stable our recommendations are with respect to fluctuations in the size and composition of the passenger flow. Simulation of delay scenarios in PANDA is done for critical transfers. In case of a waiting decision, the waiting train induces further delays in the network. PANDA's simulation framework assumes that all subsequent waiting decisions are derived from a strict application of standard waiting time rules.[1] It is quite obvious that such a simple strategy can be suboptimal. For example, passengers on a route with several transfers may only take advantage from a kept transfer if they also reach their subsequent ones. For them, an optimal decision would include "coupled decisions", where a specific waiting decision is taken jointly with one or several others (an example is given in Appendix A). Therefore, a more advanced simulation depth would be desirable, but appears to be quite challenging in a real-time setting.

Goals and contribution. We study the following questions:

1. How sensitive are waiting decisions in PANDA with respect to the given passenger flow? To this end, we want to study the following optimization problem: Determine the minimum

[1] These rules are of the form that trains of a certain train class have to wait in case of delays up to x minutes for a train of some other class. For example, an ICE train has to wait for 3 minutes for some other ICE train.

number of passengers which we have to add or to subtract from the current passenger flow such that the current decision would change. If this minimum number is small —in comparison with the number of affected passengers, then we conclude that the decision is sensitive to fluctuations. In Section 3, we show how to model this optimization problem as an integer linear program. Experimental results with realistic passenger flow data and recorded delay streams within Germany from 2015 indicate that the sensitivity of waiting decisions has to be taken into account by dispatchers.

2. What do we gain in terms of passenger punctuality by exploring coupled decisions? Or correspondingly, what do we lose by applying only standard waiting time rules in subsequent decisions? To answer these questions, we implemented a conflict tree approach for simulating more complex scenarios. Our main observation is that coupled waiting decisions improve about 6.3% of all cases.

Related work. Delay management and dispatching has been studied intensively. A first integer linear programming (ILP) formulation of delay management has been given by Schöbel [13] and extended in [14]. Several recent approaches integrated passenger rerouting into ILP formulations for delay management, for example Dollevoet et al. [5], Dollevoet and Huisman [4], Schmidt [12], and Kanai et al. [6]. Typically, they consider *offline versions*, where all delays are known before the optimization process starts. Unfortunately, the integration of the rerouting part into ILP models leads to a huge blow up of model size. For large and complex train networks with several thousands of stations and millions of passengers, the resulting ILP models cannot be solved by state-of-the-art integer programming techniques, and certainly not within very few minutes as needed in an operational setting. While the above mentioned approaches model delay management on a macroscopic operational level of detail (arrival and departure events at stations), Corman et al. [3] present a first attempt to combine detailed microscopic (i. e., block section level) delay management models with passenger routing. Dollevoet and Huisman [4] proposed and studied several fast heuristics. In particular, they introduce an iterative ILP approach which comes close to an exact ILP solution but is significantly faster. However, it is not known whether the iterative ILP approach scales well to large-scale networks. For online scenarios, Kliewer and Suhl [7] evaluate several simple dispatching rules. They work with randomly generated delay scenarios and randomly generated passenger flows while we use observed delays and more realistic passenger flows. Moreover, they work only with a subfleet of interregional trains from the Frankfurt area. Bauer and Schöbel [1] also consider various online strategies. They introduce a learning strategy based on simulations with many delay scenarios and report promising results using this strategy in experiments on artificial schedules and generated delay data.

An important aspect of train dispatching is the timing within the decision making process. Lemnian et al. [9] and Rückert et al. [11] studied the question when to decide. They showed that early rerouting is beneficial in a significant number of cases in comparison to the conservative strategy which decides as late as possible.

In this paper we focus on train dispatching for small and medium-size delays. Passenger-oriented management strategies for major disruptions, where part of the infrastructure is temporarily unavailable, have been proposed by Kroon et al. [8] and Veelenturf et al. [15]. We are not aware of any previous studies on the sensitivity of dispatching decisions.

Overview. The remainder of this work is structured as follows. In Section 2, we briefly review the train dispatching framework of PANDA. Afterwards, in Section 3, we describe our approach for analyzing the sensitivity of waiting decisions with respect to the passenger flow.

Moreover, we evaluate the results of several thousand test cases. The second theme, the impact and importance of coupled decisions, is discussed in Section 4. Finally, we conclude with a short summary and an outlook to future work (Section 5). The Appendix contains an illustrating example for coupled decisions.

2 Train Dispatching Framework

Event-activity networks and passenger flows. To model the railway schedule and passenger flow we use an *event-activity network* $\mathcal{N} = (\mathcal{V}, \mathcal{A})$, which is a directed acyclic graph with vertex set $\mathcal{V}$ and arc set $\mathcal{A}$. Each vertex represents an arrival or departure event of some train. Arcs model relations between events. We distinguish between *driving arcs*, modelling the driving of a specific train until its next stop, *waiting arcs*, modelling a train standing on a platform and *transfer arcs*, modelling the possibility for passengers to switch between trains. For more details of the event-activity network see [2] or [9]. In our model, the passenger flow is represented by directed paths in $\mathcal{N}$. Each passenger route corresponds to a path between a departure event at its origin and an arrival event at its destination. Since we are dealing with millions of passengers, we merge passengers with an identical path in the event-activity network to *passenger groups* and consider them as a kind of equivalence class. In case of delays all passengers who belong to the same group are treated in the same fashion. This approach greatly reduces the computational effort for persistently updating the passenger flow. In reality not all individual members of a passenger group will necessarily act in the same way if they have to change their original travel plans due to some kind of disturbance. Thus, more sophisticated behavioral models might be needed. However, a refinement of equivalence classes of passengers into smaller groups can easily be implemented.

Delays and critical transfers. Event nodes are equipped with time stamps: the planned event time according to schedule and the current forecast for its realization (if it lies in the future) or its realization time (if it lies in the past). In addition to the railway schedule and passenger flow we obtain delay information from Deutsche Bahn in real-time. Delay information is used to modify the event-activity network in such a way that its time stamps represent the real delay status of the railway network. Delay propagation is done as described in [10]. Therefore, the event-activity network changes its structure over time. Because of these changes it is necessary to evaluate the feasibility of each transfer. To this end, we developed a method to classify all future transfers according to the current delay situation in the event-activity network (for details see [9]). We have two different infeasible states for transfers, *critical* and *broken*. Critical means that the transfer can be maintained by a slight delay (additional to what is specified in the standard waiting time rule) of the connecting train. Broken means that the transfer can only be maintained by a large (additional) delay of the connecting train. The differentiation of the two states is important, because dispatchers should keep an eye on critical transfers and only in exceptional cases on broken ones.

Basic framework. A schematic sketch of the basic framework for train dispatching is given in Fig. 1. First, PANDA uses the schedule and passenger flow information to create an event-activity network. Every 30 seconds, our framework receives the newest delay information and updates the structure of the event-activity network accordingly. After this step the program classifies all future transfers. For all passengers affected by broken transfers PANDA determines a fastest alternative route with minimum number of transfers to their

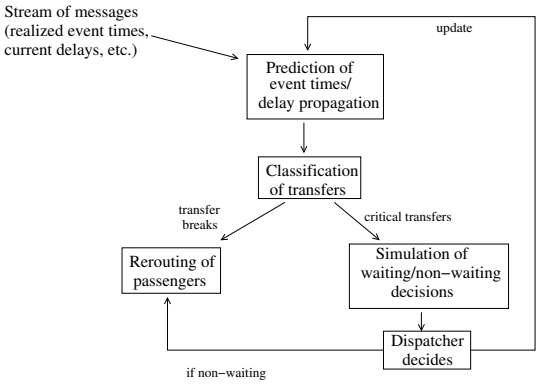

Figure 1 Schematic sketch of the train dispatching process.

destinations.[2] For all passengers using a critical transfer our software simulates a WAIT and a NO-WAIT decision and displays the evaluation of both alternatives to the dispatcher.

The chosen criteria to evaluate both decisions are listed below. If the dispatcher decides not to maintain the transfer, PANDA again determines the fastest alternative routes for all affected passengers to their destinations. If the decision is to maintain the transfer, then the dispatcher manually creates a delay for the connecting train and our framework receives this delay message in the next iteration of its dispatching process. A waiting decision for a specific transfer, say from train A to train B, may implicitly also resolve other critical transfers of trains feeding train B. In such cases our evaluation considers all affected passenger groups.

Multi-criteria objectives. We use the following criteria to evaluate the impact of waiting and non-waiting decisions on passengers. The specific choice of the following criteria is an outcome of discussions with practitioners, and the given thresholds are somewhat arbitrary. Clearly, the precise definitions are easily adaptable. Our criteria are

1. the total delay at destination (or, equivalently the average delay) over all passengers
2. number of passengers with a delay at destination < 6 minutes (regarded as "on-time")
3. number of passengers with a delay at destination ≥ 6 minutes
4. number of passengers with a delay at destination ≥ 30 minutes
5. number of passengers with a delay at destination ≥ 60 minutes
6. number of passengers with a delay at destination ≥ 120 minutes
7. number of passengers without any (acceptable) alternative.[3]

Objectives 3-6 include passengers without any acceptable alternative. We evaluate all given criteria for both alternative decisions, but consider only those passenger groups for which a difference with respect to their arrival time at the destination occurs. To provide a recommendation to dispatchers, we apply a simple *majority rule*. We recommend WAIT if the majority of criteria is in favor of waiting, and NO-WAIT if the majority of criteria is in favor of non-waiting. Otherwise, there is a tie.

[2] We only reroute passengers if necessary, i. e. if their original route has become infeasible. In some cases, other passengers may have the possibility to switch to some faster route, but such options are not supported in this context.

[3] Here we have either been unable to find any alternative route, or passengers have planned to arrive at their final destination before 02:00 a.m. on the next day, but by being rerouted to the best available alternative route they would arrive only after 04:00 a.m. on the next day. We consider an unplanned over-night trip or an extended stay at some station during night as not acceptable.

3 Sensitivity Analysis

In this section we are interested in the following optimization problem: Given a critical transfer, what is the minimum number of passengers we have to add or to subtract from the given passenger flow such that the decision of PANDA would change from waiting to non-waiting or vice versa? In our model we consider deviations from the given passenger flow in both directions. For each passenger group, the actual number of passengers can be larger or smaller than planned. A restriction of our model is that we do not consider additional passenger groups with other source-destination relations than the ones given.

Recall, that in our dispatching framework PANDA, dispositions follow a majority decision with respect to several optimization criteria. Let us assume that n passenger groups $p_1, p_2, \dots p_n$ are affected by one of the two alternative decisions (a passenger is *affected* if his route or his arrival time at the destination differs). We can identify affected passengers during the simulation by first collecting all arrival events with non-identical timestamps in the two alternatives. Second, we collect all passenger groups which have their final destination associated with these events. Moreover, we can easily keep track of rerouted passengers. Let us denote the set of affected passenger groups by $\mathcal{P}$ and let the expected number of passengers in group p_i be c_i. For each group p_i, our simulation calculates the delay at the final destination for both possibilities. In the waiting case, this delay in minutes is denoted by $W(p_i)$, whereas in the non-waiting case we write $N(p_i)$. For the ℓ-th decision criterion, let $crit^\ell(\mathcal{P})$ be the function which maps to $\{-1, 0, 1\}$, where we identify the function value 1 with the decision WAIT, the value -1 with the decision NO-WAIT, and the value 0 with a tie (NEUTRAL). For example, we can define for the first criterion (the total amount of delay at the destination) the function as

$$crit^1(\mathcal{P}) = \begin{cases} 1 & \text{if } \sum_{i=1}^n W(p_i)c_i < \sum_{i=1}^n N(p_i)c_i \\ -1 & \text{if } \sum_{i=1}^n W(p_i)c_i > \sum_{i=1}^n N(p_i)c_i \\ 0 & \text{otherwise .} \end{cases}$$

With k different criteria, we decide WAIT if $\sum_{\ell=1}^k crit^\ell(\mathcal{P}) > 0$, and NO-WAIT if this expression is negative. Otherwise, we obtain a tie.

Sensitivity problem.

Given: A critical or broken transfer for which the decision is either WAIT or NO-Wait, and a set of n affected passenger groups $\mathcal{P}$, and values $c_i, W(p_i), N(p_i)$ for each $p_i \in \mathcal{P}$.

Task: Determine numbers $\tilde{c}_i \geq 0$ such that the total deviation from the given multiplicities of passenger groups $\sum_{i=1}^n |\tilde{c}_i - c_i|$ is as small as possible and the overall decision is reversed to the opposite one.

The complexity status of the problem is unknown. At first glance, one might think that a simple greedy approach could solve the sensitivity problem. It seems natural to order the passenger groups by decreasing difference $|W(P_i) - N(P_i)|$. However, it is easy to come up with counter-examples showing that a greedy strategy based on such an order does not work. While changing the multiplicity of the group maximizing $|W(P_i) - N(P_i)|$ is most beneficial to revert the criterion total delay at destination from NO-WAIT to WAIT, its influence on other criteria is more subtle —it may or may not have an effect. A second natural greedy heuristic would order passenger groups decreasingly by the number of criteria which they can influence. But again, easy counter-examples demonstrate that a greedy approach based on this idea does not work either.

Next we provide an ILP formulation of the sensitivity problem. Its main advantage is the ease by which we can adapt the formulation to changes in the specific choice of criteria for evaluating WAIT and NO-Wait decisions. For example, one could easily add further criteria or integrate a weighting scheme to differentiate the influence of selected criteria.

3.1 ILP Formulation

We sketch the basic idea behind our ILP formulation. Let x_i^+ and x_i^- be integral variables, describing the number of passengers which are added to or subtracted from group $p_i \in \mathcal{P}$, respectively. Suppose that we have to deal with k criteria. Let $\Delta := |\sum_{j=1}^{k} crit^j(P)|$. This number Δ represents the absolute value of the difference of criteria in favor or against a WAIT decision. Hence, in order to change the overall outcome, the net change of the individual criteria must be at least $\Delta + 1$. A technical complication for our ILP formulation comes from the fact that each criterion can assume the three states WAIT, NO-WAIT, and NEUTRAL. Hence, we have to distinguish the cases that a criterion changes from WAIT to NEUTRAL or NO-WAIT, from NO-WAIT to NEUTRAL or WAIT, and from NEUTRAL to WAIT or NO-WAIT. To this end, we introduce the following three $\{0,1\}$-decision variables for each criterion. For criterion j, variable w_j denotes that the j-th criterion will vote for WAIT in an optimal solution of the sensitivity problem, whereas variable $\overline{w}_j$ means that it is in favor for NO-WAIT. The third possible outcome, a tie (NEUTRAL), is denoted by t_i.

Let $\mathcal{K}$ be the set of all criteria. With respect to the situation before solving the ILP, let $\mathcal{W}$ be the subset of criteria in favor of WAIT, $\mathcal{NW}$ the subset of criteria in favor of NO-WAIT, and $\mathcal{T}$ the remaining subset of criteria with a tie. Assuming that the current decision is NO-WAIT, we obtain the following ILP (the opposite case is very similar).

$$\min \sum_{i \in \mathcal{P}} (x_i^+ + x_i^-) \tag{1}$$

subject to

$$w_j + \overline{w}_j + t_j = 1 \quad \text{for } j \in \mathcal{K} \tag{2}$$

$$\sum_{j \in \mathcal{NW}} (2w_j + t_j) + \sum_{j \in T} (w_j - \overline{w}_j) + \sum_{j \in \mathcal{W}} (-2\overline{w}_j - t_j) \geq \Delta + 1 \tag{3}$$

$$\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - (b_j + 1)w_j - b_j t_j + M\overline{w}_j \geq 0 \quad \text{for } j \in \mathcal{NW} \tag{4}$$

$$\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - (b_j + 1)\overline{w}_j - b_j t_j + Mw_j \geq 0 \quad \text{for } j \in \mathcal{W} \tag{5}$$

$$\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) + M(t_j + \overline{w}_j) \geq 1 \quad \text{for } j \in \mathcal{T} \tag{6}$$

$$\sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - M(t_j + w_j) \leq -1 \quad \text{for } j \in \mathcal{T} \tag{7}$$

$$2 \cdot \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) - t_j + M\overline{w}_j \geq 0 \quad \text{for } j \in \mathcal{T} \tag{8}$$

$$-2 \cdot \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-) + t_j + Mw_j \geq 0 \quad \text{for } j \in \mathcal{T} \tag{9}$$

$$x_i^- \leq c_i \quad \text{for } i \in \mathcal{P} \tag{10}$$

$$x_i^+, x_i^- \in \mathbb{N}_0 \quad \text{for } i \in \mathcal{P} \tag{11}$$

$$w_j, \overline{w}_j, t_j \in \{0,1\} \quad \text{for } j \in \mathcal{K}, \tag{12}$$

where M is a sufficiently large constant, and the coefficients $a_{ij} \in \mathbb{Z}$ denote the contribution of passenger group $i \in \mathcal{P}$ to criterion $j \in \mathcal{K}$, and the coefficients $b_j \in \mathbb{Z}$ the amount by which the current evaluation of criterion $j \in \mathcal{W} \cup \mathcal{NW}$ has to be changed in order switch this criterion from WAIT or NO-WAIT to NEUTRAL.

The objective function expresses that we want to minimize the necessary change. In an optimal solution, at most one of each pair of variables x_i^+, x_i^- can be strictly positive. Equality (2) in combination with the 0-1-variable bounds in (12) ensures that exactly one of the three possible states (WAIT, NO-WAIT, NEUTRAL) is chosen for each criterion. In Inequality (3), the left-hand-side sums up the total change of criteria. To fulfill the inequality, the sum must be large enough to change the decision from NO-WAIT to WAIT. Inequalities (4)-(9) link the change in passenger flow to the different criteria. We use a "big-M" formulation to ensure that we can always fulfill all of these inequalities. The expression $z_j = \sum_{i \in \mathcal{P}} a_{ij}(x_i^+ - x_i^-)$ measures the effect of the passenger flow change on criterion $j \in K$. For $j \in \mathcal{NW}$ (the j-th criterion is currently in favor of NO-WAIT), we can fulfill Inequality (4) with $w_j = 1$ (or $t_j = 1$) if z_j is large enough to reverse the criterion to WAIT (or to NEUTRAL, respectively). Otherwise, we can always choose $\overline{w}_j = 1$. Since it helps to fulfill Inequality (3), we may safely assume that an optimal solution prefers setting $w_j = 1$ over $t_j = 1$ and the latter over $\overline{w}_j = 1$. For $j \in \mathcal{W}$, Inequality (5) works analogously. Inequalities (6)-(9) together model the case that the current state of a criterion is NEUTRAL. Here, a case analysis shows that $z_j > 0$ implies $w_j = 1$, $z_j < 0$ implies $\overline{w}_j = 1$, and $z_j = 0$ implies $t_j = 0$. Inequality (10) ensures for each group that we cannot subtract more passengers than we currently have.

3.2 Experiments

Experimental Setup. We use the German train schedule of 2015 including all long-distance and regional trains. Overall, we have about 66000 trains and a million events per day. In addition to the schedule we obtained realistic passenger flow data from Deutsche Bahn. The used model contains about 3.3 million passengers on roughly 320 000 different routes per day. This passenger flow includes only passengers which use at least one long-distance train. With respect to our flow about 28000 different transfers are used by passengers every day. For our evaluation we used recorded data for actual delays of eight weekdays in June and October 2015. Every single test day is simulated by spreading the recorded delays into the network. For each detected critical or broken transfer we simulate a PANDA decision 15 minutes before the connecting train is scheduled to depart. If the evaluation suggests either to wait or not to wait we calculate the minimal number of passengers to change the suggested strategy. We solved the corresponding ILPs by using the non-commercial SCIP Optimization Suite[4] in version 3.2.1 with SoPlex 2.2.1 as the ILP-solver. Overall, we examined 73486 many PANDA decisions.

Experimental Results. To allow a comparison between different scenarios, we normalize the necessary total passenger change (i. e., the value of the optimal ILP solution) by the number of affected passengers (more precisely, by the number of passengers which are affected differently by the two alternative decisions). This gives us a kind of reliability measure

$$rel = \frac{\text{total passenger change}}{\#\text{affected passengers}}.$$

[4] `http://scip.zib.de`

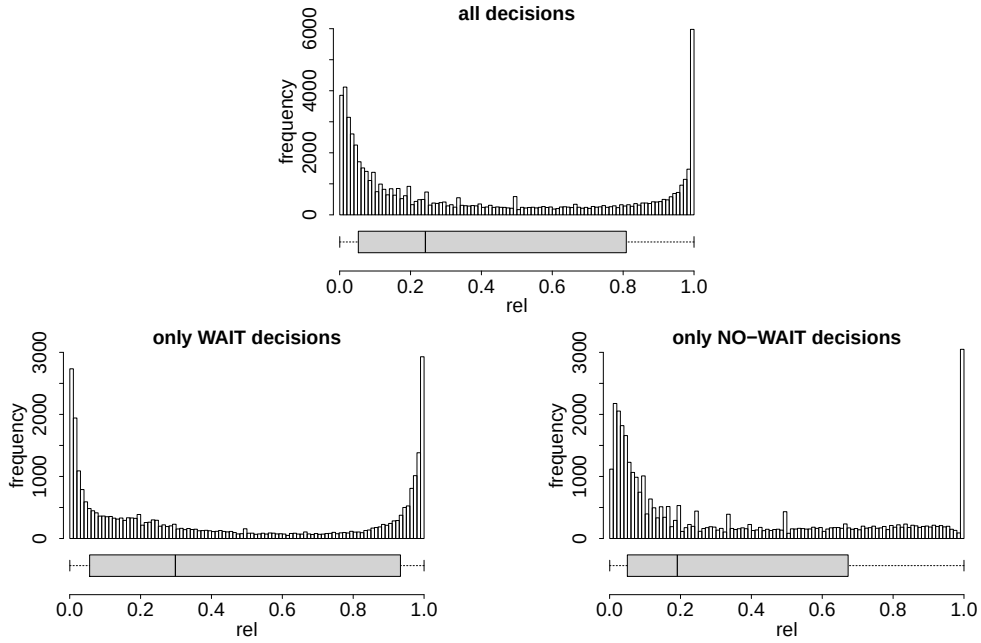

Figure 2 Top: sensitivity for all decisions: on average about 40% of the passengers are needed to change the decision. The median is about 24%. Left: sensitivity of WAIT-decisions: on average about 45% of the passengers are needed to change the decision. The median is about 30%. Right: sensitivity of NO-WAIT-decisions: on average about 36% of the passengers are needed to change the decision. The median is about 19%.

Since the optimal solution will never change the passenger flow by more than removing all existing passengers, we clearly have $0 \leq rel \leq 1$. The larger the value of rel, the more robust is the corresponding decision.

In the top part of Figure 2, a histogram shows the empirical distribution of the reliability measure rel based on all considered cases. This distribution appears to be U-shaped. A significant number of cases turns out to be highly sensitive, and another significant portion of all cases is quite robust. The mean of rel is .4, that is, on average we need about 40% of the affected passengers to change the suggested decision. Note that the distribution is skewed, and that the median is only .24. In the lower part of Figure 2, we distinguish between WAIT (left) and NO-WAIT decisions (right). It is an interesting observation that WAIT decisions turn out to be more robust than NO-WAIT decisions on average. The median of the sensitivity measure is .29 in case WAIT, in comparison to .19 in case NO-WAIT.

4 Coupled Decisions

We are now going to study the possible benefit of coupled waiting decisions. For each potential waiting decision of a critical or broken transfer, we do this evaluation in two steps. First, we recursively build up a conflict tree structure representing the dependencies of the given waiting decision with other subsequent decisions. In a second phase, we evaluate the impact of every choice of coupled waiting decisions by enumerating subtrees of the conflict tree. Note that this approach is only meant for the purpose of an a posteriori evaluation. Therefore, running times are neglected.

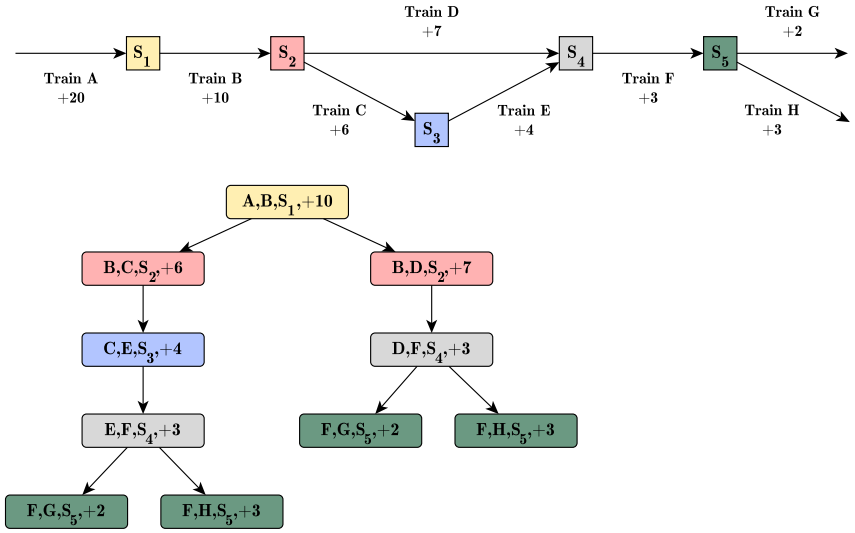

Figure 3 Example scenario of coupled decisions. Upper part: Involved trains and the induced delay if the transfer from its feeder train is kept. Lower part: the corresponding conflict tree. For each vertex, we denote the corresponding transfer and waiting decision by the feeder train, the departing train, the station, and the required artificial delay to keep the transfer.

4.1 The Conflict Tree

Given a critical or broken transfer tr, its associated *conflict tree* $T_{tr} = (V(T_{tr}), E(T_{tr}))$ is defined recursively. Every vertex of T_{tr} represents a critical or broken transfer in the underlying event-activity network. The root of conflict tree T_{tr} represents the initial critical/broken transfer tr. A conflict tree consists of a single vertex if waiting for the feeder train does not make any other transfers critical or broken. Otherwise, we obtain non-trivial conflict trees. For each tree vertex $v \in V(T_{tr})$, its children correspond exactly to all those transfers which are critical or broken under the condition that the transfers corresponding to v and all its predecessors in the tree are kept.

Keeping a critical or broken transfer tr' means, we have to delay the corresponding departure event such that passengers have enough time to reach the departing train, say by $d(tr')$ minutes. Such an artificial delay has to be propagated through the event-activity network. Propagated delays may influence other transfers in all directions. They may newly create, worsen or even maintain following critical or broken transfers. Every non-root vertex of the conflict tree represents a critical or broken transfer induced by a WAIT decision for some other transfer higher up in the tree. Since NO-WAIT decisions do not alter the delay scenario, they do not lead to follow-up conflicts. We would like to point out that the same transfer can be represented several times within a conflict tree. See for example Fig. 3 where all leave nodes occur twice.

Conflict trees may have a self-similarity or fractal property as shown in Fig. 4. Self-similarity/fractal means that subtrees and their associated transfers are similar to other subtrees. In the example presented in Fig. 4 the subtree of vertex $v2$ is similar to the subtree of vertex $v1$ (without $v2$). However, the necessary delays to maintain the individual transfers can be different, because of the different delay situation in both subtrees. For instance, $v3$ and $v4$ are critical/broken transfers by spreading the delay $d(v1)$ into the network. Nevertheless, $v3'$ and $v4'$ are the same critical/broken transfers, but by spreading the delays $d(v1)$ and

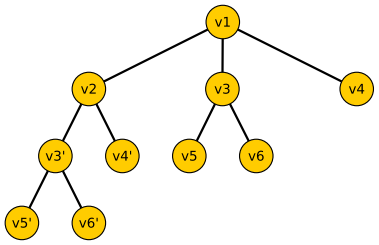

Figure 4 An example of a conflict tree with a self-similarity property. The subtree of vertex $v2$ is similar to the subtree of vertex $v1$ (without $v2$). The vertices $v3$ to $v6$ and $v3'$ to $v6'$ correspond to the same transfers, but the necessary delays to maintain the transfers might be different.

then $d(v2)$ into the network. For a large-scale network like that of Germany, this property can lead to very large conflict trees with over several millions of vertices.

Creation of a Conflict Tree. Our algorithm to create a conflict tree is similar to breadth-first search. We start with an empty queue $\mathcal{Q}$ and push the root vertex with the initial critical/broken transfer into it. As long as there are vertices in the queue, we explore its first element (and then perform a dequeue operation). The current vertex v has to be maintained by propagating an artificial delay $d(v)$ in the underlying network $\mathcal{N}$. All thereby induced critical/broken transfers are collected and then inserted into the tree as well as into the queue. Let us consider the example given in Figure 4. We start with vertex $v1$ and spread the delay $d(v1)$ into the network $\mathcal{N}$. Then, we collect the induced conflicts $v2, v3$ and $v4$. Next we continue with vertex $v2$ and spread the delay $d(v2)$ into the network. We also collect the subsequent conflicts $v3'$ and $v4'$. Now we would like to process $v3$, but the current state of the network is modified by the two delays $d(v1)$ and $d(v2)$. The delay $d(v2)$ is unnecessary to measure the impact of delay $d(v3)$. Therefore, we have to re-establish a valid state of the network before spreading the delay of the current vertex. For sake of simplicity, we remove *all* artificial delays directly after we inserted the induced subsequent critical/broken transfers into the queue and re-propagate all artificial delays from the root vertex to the predecessor of the current vertex v directly before spreading the delay. Thereby, we always ensure that the current state of the underlying network is valid.

Evaluation of a Conflict Tree. For each vertex in the tree we have a binary decision variable to model that this transfer will be maintained (one) or not (zero). We can interpret these $|V = V(T_{tr})|$ many binary decision variables as a $|V|$ bit long variable x. This variable x can theoretically attain $2^{|V|}$ different values, but not all of these values are feasible. For instance, if the root vertex is assigned with a zero, then there are no following conflicts, therefore the remaining $|V| - 1$ bits have to be set to zero. In general, a $|V|$ bit long variable x is a *feasible* configuration for a conflict tree, if and only if for every bit i that is set to one all bits corresponding to the path from the root to the predecessor of v_i in the conflict tree are also set to one.

The evaluation algorithm has two phases. In the first phase we determine the set of all affected passenger groups. To do so, we simulate all feasible coupled decisions successively and collect from all simulated decisions the affected passenger groups. Note that we have to re-establish the original state of the event-activity network after every simulation step. Working with the set of affected passengers is necessary to have an unbiased comparison between the impact of all simulated feasible decisions on the passenger flow. In this phase no

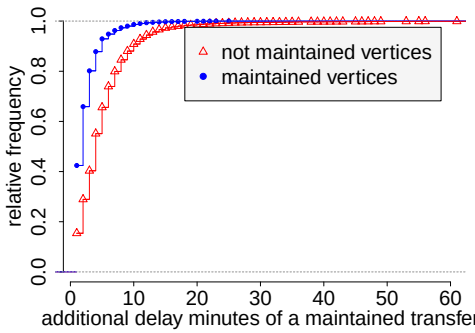

Figure 5 Cumulative distribution functions of the additional delay minutes for both maintained and not maintained vertices of all conflict trees.

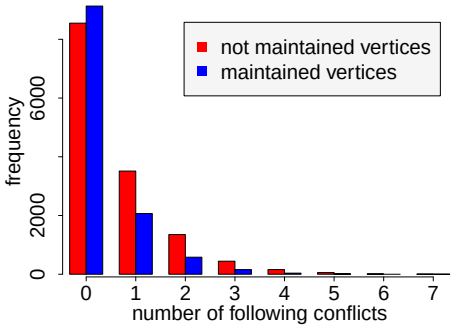

Figure 6 Distribution function of the number of follow-up conflicts for both maintained and not maintained vertices of all conflict trees.

passenger flow adjustments are done. In the second phase we again iterate over and simulate all feasible coupled decisions successively, but in each step we measure and store the impact of the injected delays on the previously collected passenger groups. The passenger groups may have to be rerouted at this point if their route is not feasible any more. After these two steps we can compare all feasible coupled decisions with each other in an unbiased way. Finally, we are able to evaluate the impact on the passenger flow of all feasible coupled decisions. For each scenario we compute objective values for all seven PANDA criteria. As before, we compare two solution vectors by counting the number of criteria where one solution is strictly better than the other. Hence, scenario A is considered as better as scenario B if the majority of criteria is in favor of scenario A.

4.2 Experiments

Experimental Setup. We use the same German train schedule for the evaluation of the benefit of coupled waiting decisions as in the previously described experiment. For every critical/broken transfer we calculate the conflict tree 15 minutes in advance of the scheduled event time and evaluate the impact of all feasible coupled decisions on the passenger flow. Because of the high computational effort to determine all feasible coupled decisions we focus only on conflict trees with at most 10 vertices/conflicts (at most 1024 different coupled decisions). Note that the larger the tree becomes the less likely it is that coupling is preferable. Finally, we collect all evaluations and compare them with pure NO-WAIT and WAIT decisions. By this process we obtained 20920 different conflict trees.

Experimental Results. For these 20920 conflict trees we have found 4941 cases (about 23.61%) where coupled waiting decisions are better than single WAIT-decisions. Furthermore, there are 2982 cases (about 14.25%) in which the coupled waiting decisions are better than NO-WAIT-decisions. However, there are only 1319 cases (about 6.3%) where coupled waiting decisions are better than both WAIT- and NO-WAIT-decisions.

Next we are interested to understand under which circumstances coupled decisions are preferable. In the cases where coupled waiting decisions are at least better than WAIT- or NO-WAIT-decisions we collect all maintained non-root vertices (about 12000). Similarly, we also collect all not maintained non-root vertices of all remaining scenarios (about 14000 vertices). For both sets of vertices we consider several properties of its members. These

Table 1 The average change for three criteria by applying coupled waiting decisions in comparison with standard single WAIT/NO-WAIT decisions.

criteria	benefit of coupled waiting decisions
total arrival delay	-3.02%
# passengers with $\geq$ 60 min. delay	2.34%
# passengers with $\geq$ 120 min. delay	.58%

properties are for instance: the number of minutes required to maintain the corresponding transfer and the number of its children in the conflict tree. Figure 5 shows that it is more likely to have a maintained transfer if the additional delay minutes are quite small. In addition, Figure 6 shows that it is more likely for a vertex to become a maintained transfer if it is a leaf in the decision tree. If a vertex has at least one child it is about 15% more likely to be a non-maintained transfer. We conclude that a heuristic pruning scheme should preferably explore vertices which require a small extra delay or those which induce no follow-up conflicts (that is, leaves in the decision tree).

To measure the benefit of coupled waiting decisions we compare the standard single WAIT/NO-WAIT decisions with the best solution we can obtain for either WAIT, NO-WAIT, or a coupled decision according to three different criteria. As shown in Table 1 the coupled waiting decisions have slightly worsened the total arrival delay by about 3%. Nevertheless, the number of passengers with an arrival delay of at least 60 or at least 120 minutes could be reduced by about 2% respectively by .58%. Thus, the overall benefit of coupled decisions is mixed, but the improvements for passengers with large delays should outweigh their slightly larger average delay.

5 Summary and Future Work

In this paper we have discussed two enhancements of the dispatching framework provided by PANDA. First, we showed how to provide sensitivity information for dispatching recommendations with respect to fluctuations within the passenger flow. For each critical transfer, we can tell whether our waiting or non-waiting recommendation is stable under slight changes of the passenger flow. Our main finding is that the overall distribution of the sensitivity is U-shaped. That means, we observe a significant fraction of cases that are either very stable or very unstable. We conclude that the knowledge the specific sensitivity of a critical WAIT/NO-WAIT decision is highly valuable for the decision making process. If the sensitivity is low, an automatized decision might be possible, whereas a high sensitivity indicates that a human dispatcher is required to take a closer look into the pros and cons of the decision in question. Future work should also study a second dimension of uncertainty in the given data: How sensitive are waiting decisions with respect to delay predictions?

Second, we explored the value of coupled decision making which extends the analysis of critical transfers. We learned that the large extra work spent in exploring larger parts of conflict trees only pays off in relatively rare cases. In most cases, just exploring the root node and working with standard waiting time rules for all other nodes of the conflict tree already yields an optimal solution. As a next step, we would like to exploit these observations to develop heuristic rules for pruning conflict trees. Up to now, the conflict tree part of our prototype has not been optimized for efficiency. Hence, we will work on speeding it up to meet the requirements of real-time dispatching.

References

1 Reinhard Bauer and Anita Schöbel. Rules of thumb — practical online strategies for delay management. *Public Transport*, 6:85–105, 2014.

2 Annabell Berger, Christian Blaar, Andreas Gebhardt, Matthias Müller-Hannemann, and Mathias Schnee. Passenger flow-oriented train disposition. In C. Demetrescu and M. M. Halldórsson, editors, *Proceedings of the 19th Annual European Symposium on Algorithms (ESA)*, volume 6942 of *LNCS*, pages 227–238. Springer, 2011.

3 Francesco Corman, Dario Pacciarelli, Andrea D'Ariano, and Marcella Samà. Railway traffic rescheduling with minimization of passengers' discomfort. In *Computational Logistics, ICCL 2015*, volume 9335 of *LNCS*, pages 602–616. Springer, 2015.

4 Twan Dollevoet and Dennis Huisman. Fast heuristics for delay management with passenger rerouting. *Public Transport*, 6:67–84, 2014.

5 Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1):74–89, 2012.

6 Satoshi Kanai, Koichi Shiina, Shingo Harada, and Norio Tomii. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. *Journal of Rail Transport Planning & Management*, 1:25 – 37, 2011.

7 Natalia Kliewer and Leena Suhl. A note on the online nature of the railway delay management problem. *Networks*, 57:28–37, 2011.

8 Leo G. Kroon, Gabor Maróti, and Lars K. Nielsen. Rescheduling of railway rolling stock with dynamic passenger flows. *Transportation Science*, 49:165–184, 2015.

9 Martin Lemnian, Ralf Rückert, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. Timing of train disposition: Towards early passenger rerouting in case of delays. In Stefan Funke and Matús Mihalák, editors, *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2014*, volume 42 of *OASICS*, pages 122–137. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

10 Matthias Müller-Hannemann and Mathias Schnee. Efficient timetable information in the presence of delays. In R. Ahuja, R.-H. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 249–272. Springer, 2009.

11 Ralf Rückert, Martin Lemnian, Steffen Rechner, Christoph Blendinger, and Matthias Müller-Hannemann. PANDA: A software tool for improved train dispatching with focus on passenger flows. In *Proceedings of CASPT 2015 (Conference on Advanced Systems in Public Transport), Rotterdam*. 2015. URL: `www.caspt.org/proceedings/paper90.pdf`.

12 Marie Schmidt. Simultaneous optimization of delay management decisions and passenger routes. *Public Transport*, 5:125–147, 2013.

13 Anita Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

14 Anita Schöbel. Integer programming approaches for solving the delay management problem. In F. Geraets, L. Kroon, A. Schoebel, D. Wagner, and C. Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *LNCS*, pages 145–170. Springer, 2007.

15 Lucas P. Veelenturf, Leo G. Kroon, and Gábor Maróti. Passenger oriented railway disruption management by adapting timetables and rolling stock schedules. In *10th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014)*, pages 11–34. 2014.

A Example: Coupled Decisions

In Figures 7 – 12 we provide a small example to illustrate a typical scenario where coupling of decisions is reasonable. The first figure shows the planned scenario according to schedule

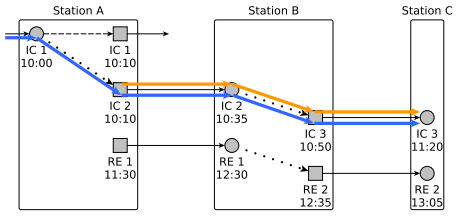

Figure 7 Planned scenario.

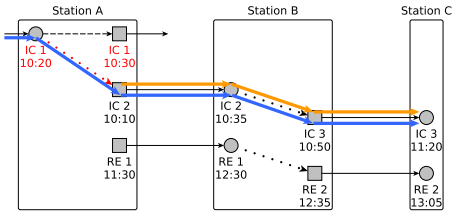

Figure 8 Train IC 1 delayed by 20 minutes.

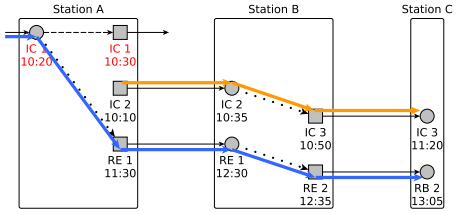

Figure 9 NON-WAITING case: The transfer from IC 1 to IC 2 breaks. Passengers on this transfer have to be rerouted to later trains.

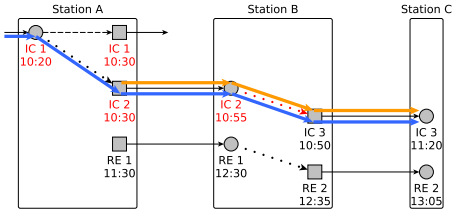

Figure 10 WAITING case: The transfer from IC 1 to IC 2 is maintained. But as a side effect the transfer from IC 2 to IC 3 at station B becomes critical.

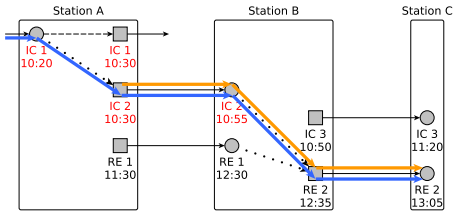

Figure 11 The transfer from IC 2 to IC 3 breaks. Several passenger groups are rerouted.

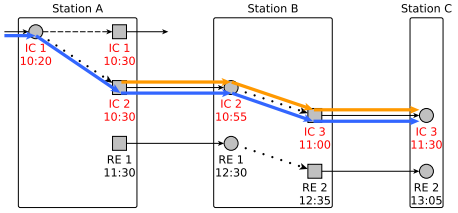

Figure 12 Coupled waiting decision: Both transfers (IC 1 to IC 2 and IC 2 to IC 3) are kept. All passenger groups stay on their original route.

with two passenger groups (their travel paths are shown in blue and orange, respectively). Next, we assume that train IC 1 is delayed by 20 minutes. This makes the transfer from IC 1 to IC 2 for one passenger group critical. If IC 2 does not wait, passengers on this transfer have to be rerouted. If, however, the transfer from IC 1 to IC 2 is maintained, the late departure of IC 2 causes another critical transfer from IC 2 to IC 3. If this transfer is not maintained, the situation becomes even worse, since both passenger groups have to be rerouted. Here we see a prototypal use-case for coupled decisions: if both transfers are kept, all passengers can stay on their original route and their total delay is minimized.