IP-based Techniques for Delay Management with Priority Decisions [⋆]

Michael Schachtebeck and Anita Schöbel

Institute for Numerical and Applied Mathematics,
Georg-August University, Göttingen, Germany.
{schachte,schoebel}@math.uni-goettingen.de

Abstract. *Delay management* is an important issue in the daily operations of any railway company. The task is to update the planned timetable to a *disposition timetable* in such a way that the inconvenience for the passengers is as small as possible. The two main decisions that have to be made in this respect are the *wait-depart decisions* to decide which connections should be maintained in case of delays and the *priority decisions* that determine the order in which trains are allowed to pass a specific piece of track. They later are necessary in the capacitated case due to the limited capacity of the track system and are crucial to ensure that the headways between different trains are respected and that single-track traffic is routed correctly. While the wait-depart decisions have been intensively studied in literature (e.g. [Sch06,Gat07]), the priority decisions in the capacitated case have been neglected so far in delay management optimization models.

In the current paper, we add the priority decisions to the integer programming formulation of the delay management problem and are hence able to deal with the capacitated case. Unfortunately, these constraints are disjunctive constraints that make the resulting event-activity network more dense and destroy the property that it does not contain any directed cycle. Nevertheless, we are able to derive reduction techniques for the network which enable us to extend the formulation of the never-meet property from the uncapacitated delay management problem to the capacitated case. We then use our results to derive exact and heuristic solution procedures for solving the delay management problem.

The results of the algorithms are evaluated both from a theoretical and a numerical point of view. The latter has been done within a case study using the railway network in the region of Harz, Germany.

1 Introduction

The delay management problem deals with (small) source delays of a railway system as they occur in the daily operational business of any public transportation company. In case of such delays, the scheduled timetable is not feasible any more and has to be updated to a *disposition timetable*. The main question which has been treated in the literature so far is to decide which trains should wait for delayed feeder trains and which trains better depart on time (*wait-depart decisions*).

A first integer programming formulation for the uncapacitated delay management problem has been given in [Sch01] and has been further developed in [GHL08,Sch07b], see also [Sch06] for an overview about various models. The complexity of the problem has been investigated in [GJPS05,GGJ$^+$04] where it turns out that the problem is NP-hard even in very special cases. The online version of the problem has been studied in [GJPW07,Gat07]. In [BHLS07], it was shown that the online version of the uncapacitated delay management problem is PSPACE-hard. Further publications about delay management include a model in the context of max-plus-algebra ([RdVM98,Gov98]), a formulation as discrete time-cost tradeoff problem ([GS07]) and simulation approaches ([SM99,SMBG01])

[⋆] This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

However, these studies neglect the limited capacity of the track system while dealing with delay management. Adding these constraints, the problem becomes significantly harder to solve. Some first ideas on how to model these constraints in the context of delay management have been presented in [Sch07a]. Capacity constraints are also taken into account in a real-world application studied within the project *DisKon* supported by Deutsche Bahn (see [BGJ+05]). Here, the following setting to apply delay management in practice is suggested: In a first step, a macroscopic approach deals with the wait-depart decisions, while a second step ensures feasibility within a microscopic model. This is done by postponing departures until the track to be used is available. It may however yield rather bad solutions.

In the following, we will for the first time analyze the integer programming formulation of the delay management problem for the capacitated case.

The remainder of the paper is structured as follows. In Section 2 we present an integrated integer programming model including the priority decisions and hence respecting the limited capacity of the track system. We analyze the formulation, present reduction techniques and extend the never-meet property in Section 3. We then discuss in which cases the problem can be solved exactly using the software Xpress. Four heuristic approaches are described and analyzed in Section 4. A numerical evaluation of these approaches also is presented in Section 4. We finally conclude the paper mentioning ideas for further research.

2 Integer Programming Formulation

The uncapacitated delay management problem is defined as follows: Given the public transportation network $PTN = (V, E)$ (consisting of the set V of stations and the set E of direct links between stations), the set $\mathcal{F}$ of trains, a set of connections and some source delays, decide which connections should be maintained and which connections should be dropped such that the average delay of a passenger at his final destination is minimal. This problem was first introduced in [Sch01].

In this paper, we take the limited capacity of the tracks into account to obtain the *delay management problem with capacity constraints*. This means we also have to decide which train should drive first if two or more trains use the same piece of infrastructure (for example on single-track lines or when two consecutive trains use the same track in the same direction). Note that it can be better to change the originally scheduled order of the trains to reduce the delay. A first model of this problem has been introduced in [Sch07a]. Here we present and analyze its formulation as integer program.

To this end, we first introduce the corresponding *event-activity network* which is a directed graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ (see [Nac98,Sch07b]). $\mathcal{E}$ consists of arrival and departure events $\mathcal{E}_{\text{arr}}$ and $\mathcal{E}_{\text{dep}}$, respectively. A timetable $\pi \in \mathbb{N}^{|\mathcal{E}|}$ assigns a time π_i to each event $i \in \mathcal{E}$. If a delay occurs, we need to update the given timetable π to a so called *disposition timetable* $x \in \mathbb{N}^{|\mathcal{E}|}$. To present the constraints that have to be satisfied by a (disposition) timetable, we need the following four different types of activities, $\mathcal{A} = \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{change}} \cup \mathcal{A}_{\text{head}}$:

- driving activities $\mathcal{A}_{\text{drive}} \subset \mathcal{E}_{\text{dep}} \times \mathcal{E}_{\text{arr}}$ modeling the driving of a train between two consecutive stations (including turn-around edges),
- waiting activities $\mathcal{A}_{\text{wait}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$,
- changing activities $\mathcal{A}_{\text{change}} \subset \mathcal{E}_{\text{arr}} \times \mathcal{E}_{\text{dep}}$ which are used to model transfers from one train to another one, and
- headway activities $\mathcal{A}_{\text{head}} \subset \mathcal{E}_{\text{dep}} \times \mathcal{E}_{\text{dep}}$ which model the limited capacity of the track system.

If two events $i, j \in \mathcal{E}$ are connected by an activity $(i, j) \in \mathcal{A}$, then event i has to be performed before event j can take place. In particular, each activity $a = (i, j) \in \mathcal{A}$ has assigned a duration. If a is a driving, waiting, or changing activity, this duration is denoted by $L_a = L_{ij}$, and we require

$$x_j - x_i \geq L_{ij}.$$

The headway activities, on the other hand, appear in pairs: if $(i,j) \in \mathcal{A}_{\text{head}}$, then $(j,i) \in \mathcal{A}_{\text{head}}$, too. This is used to model the disjunctive constraints $x_j - x_i \geq L_{ij}$ or $x_i - x_j \geq L_{ji}$. The goal is to choose exactly one activity of each such pair and to respect the resulting constraint. This means that a priority decision has to be made.

In order to present the integer programming formulation, we need some more parameters:

First, we allow two types of source delays: The first is a delay d_i at an event $i \in \mathcal{E}$ (e.g. a driver coming too late to his duty), which refers to a fixed point of time, such that $x_i \geq \pi_i + d_i$ is required. The second is a delay $d_a = d_{ij}$ which increases the duration of an activity $a = (i,j) \in \mathcal{A}$, e.g. an increase of traveling time between two stations due to construction work. Such a delay d_a has to be added to the duration L_a of activity a. If an activity has no source delay (this is, for example, the case for all headway activities as we do not allow activity delays on headway activities), we assume $d_a = 0$ to simplify the notation.

Moreover, let w_i be the number of passengers getting off at event $i \in \mathcal{E}$ and w_a be the number of passengers who want to use a connection $a \in \mathcal{A}_{\text{change}}$. Throughout this paper, we assume $w_a > 0$ for all $a \in \mathcal{A}_{\text{change}}$ (otherwise, nobody uses the connection, so it can be removed from the network). We further assume that all lines have a common period T (this assumption can easily be relaxed by introducing periods T_a for all changing activities $a \in \mathcal{A}_{\text{change}}$).

To model the wait-depart decisions, we introduce binary variables

$$z_a = \begin{cases} 0 & \text{if changing activity } a \text{ is maintained} \\ 1 & \text{otherwise} \end{cases} \qquad g_{ij} = \begin{cases} 0 & \text{if event } i \text{ takes place before event } j \\ 1 & \text{otherwise} \end{cases}$$

for all changing activities $a \in \mathcal{A}_{\text{change}}$ and for all headway activities $(i,j) \in \mathcal{A}_{\text{head}}$. The integer programming formulation reads as follows:

$$(\mathbf{DM}) \;\; \min f(x,z,g) = \sum_{i \in \mathcal{E}_{\text{arr}}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T \tag{1}$$

such that

$$x_i \geq \pi_i + d_i \quad \forall i \in \mathcal{E} \tag{2}$$
$$x_j - x_i \geq L_a + d_a \quad \forall a = (i,j) \in \mathcal{A}_{\text{nice}} := \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}} \tag{3}$$
$$M z_a + x_j - x_i \geq L_a \quad \forall a = (i,j) \in \mathcal{A}_{\text{change}} \tag{4}$$
$$M g_{ij} + x_j - x_i \geq L_{ij} \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{5}$$
$$x_i \in \mathbb{N} \quad \forall i \in \mathcal{E} \tag{6}$$
$$z_a \in \{0,1\} \quad \forall a \in \mathcal{A}_{\text{change}} \tag{7}$$
$$g_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{8}$$
$$g_{ij} + g_{ji} = 1 \quad \forall (i,j) \in \mathcal{A}_{\text{head}} \tag{9}$$

where M is a constant which is “large enough”. We will show in Corollary 2 that M can indeed be chosen finitely beforehand. But let us first explain the meaning of the objective function and of the constraints:

In the objective function, we minimize the sum of all delays passengers have when starting their trips or at their final destinations plus the sum of all missed connections. It approximates the sum of all delays over all customers. Furthermore, note that any optimal solution of this program is a Pareto solution with respect to the two objective functions *minimize the delay over all vehicles* and *minimize the number of missed connections.*

Constraints (3) make sure that the delay is passed on correctly along waiting and driving activities. (4) and (5) do the same for changing activities that are maintained and for the headway activities which should be respected. Constraint (9) ensures that exactly one of each pair of headway constraints is respected.

Relaxing all constraints modeling the limited capacity of the tracks yields the uncapacitated delay management problem:

$$(\mathbf{UDM})\ \min f(x,z) = \sum_{i \in \mathcal{E}_{\text{arr}}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T$$

$$\text{such that (2), (3), (4), (6), (7) are satisfied.}$$

Let us denote the objective value of the optimal solution of (DM) by F^{DM} and let F^{UDM} be the objective value of the corresponding instance of (UDM). Since (UDM) is a relaxation of (DM), we obtain $F^{\text{UDM}} \leq F^{\text{DM}}$.

Later on, we will fix the priority variables heuristically and treat the resulting headway constraints as the constraints in (3). We hence define for some set $\mathcal{A}_{\text{fix}} \subseteq \mathcal{A}_{\text{head}}$ the problem

$$(\mathbf{UDM}(\mathcal{A}_{\text{fix}}))\ \min f(x,z) = \sum_{i \in \mathcal{E}_{\text{arr}}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{\text{change}}} z_a w_a T$$

such that

$$x_j - x_i \geq L_a + d_a \quad \forall a = (i,j) \in \mathcal{A}_{\text{fix}} \tag{10}$$

$$\text{and such that (2), (3), (4), (6), (7) are satisfied.}$$

Note that UDM($\mathcal{A}_{\text{fix}}$) yields a *feasible* solution of (DM) if $\mathcal{A}_{\text{fix}} = \{(i,j) \in \mathcal{A}_{\text{head}} : g_{ij} = 0\}$ for some $g \in \{0,1\}^{|\mathcal{A}_{\text{head}}|}$ which satisfies (9) (and provided that UDM($\mathcal{A}_{\text{fix}}$) is feasible). In this case we obtain $F^{\text{UDM}} \leq F^{\text{DM}} \leq f^{\text{UDM}(\mathcal{A}_{\text{fix}})}$.

We may also fix the variables z_a and g_{ij} and obtain $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\ \} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : g_{ij} = 0\ \}$. Determining the remaining variables x_i in (DM) then reduces to a simple project planning problem:

$$(\mathbf{PP}(\mathcal{A}_{\text{fix}}))\ \min f(x) = \sum_{i \in \mathcal{E}_{\text{arr}}} w_i(x_i - \pi_i)$$

$$\text{such that (2), (3), (10), and (6) are satisfied.}$$

The version of (DM) in which $\mathcal{A}_{\text{change}} = \emptyset$ has been shown to be NP-complete in [CS07]. This yields NP-completeness of (DM). However, (PP($\mathcal{A}_{\text{fix}}$)) can be solved in polynomial time, e.g. by applying the forward phase of the critical path method (CPM) of project planning (see [Elm77]) as follows: We first sort $\mathcal{E} = \{i_1, \ldots, i_{|\mathcal{E}|}\}$ topologically and obtain an order $\prec$. Then we set

$$\tilde{x}_{i_1} := \pi_{i_1} + d_{i_1} \tag{11}$$

$$\text{for all } k \in \{i_2, \ldots, i_{|\mathcal{E}|}\} : \tilde{x}_k := \max \Big\{ \pi_k + d_k, \max_{a=(i,k) \in \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\text{nice}}} \tilde{x}_i + L_a + d_a \Big\}. \tag{12}$$

We now come back to the integer programming formulation of (DM) and show that M is finite and can be chosen beforehand for any instance of (DM). To this end, we first give an upper bound on the maximum time of each single event in an optimal disposition timetable. We denote the slack time of an activity $a = (i,j)$ as

$$s_a = \pi_j - \pi_i - L_a,$$

i.e. s_a gives the buffer time included in the scheduled duration of the activity.

Theorem 1. *Let an instance of (DM) be given and let*

$$D := \max_{i \in \mathcal{E}} d_i + \sum_{a \in \mathcal{A}_{\text{nice}}} (d_a - s_a)_+ + \sum_{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i > \pi_j} \pi_i - \pi_j + L_{ij}. \tag{13}$$

Then there exists an optimal solution (x, z, g) of (DM) such that $x_k \leq \pi_k + D$ for all $k \in \mathcal{E}$.

Proof. We show the following stronger statement: For any feasible solution $(\bar{x}, z, g)$ of (DM), there exists a feasible solution $(\tilde{x}, z, g)$ with $\tilde{x}_k \leq \bar{x}_k$ (hence $f(\tilde{x}, z, g) \leq f(\bar{x}, z, g)$) that fulfills $\tilde{x}_k \leq \pi_k + D$ for each $k \in \mathcal{E}$.

Given $(\bar{x}, z, g)$, let $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : g_{ij} = 0\}$. As $(\bar{x}, z, g)$ is a feasible solution, $\mathcal{N}' := (\mathcal{E}, \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}})$ does not contain any directed circle, so we sort $\mathcal{E} = \{i_1, \ldots, i_{|\mathcal{E}|}\}$ topologically. We now solve $\text{TT}(\mathcal{A}_{\text{fix}})$ optimally and denote the solution obtained by $\tilde{x}$ (see (11) and (12)). Then $(\tilde{x}, z, g)$ is a feasible timetable satisfying $\tilde{x}_k \leq \bar{x}_k$.

Claim: For each $k \in \mathcal{E}$ we have $\tilde{x}_k \;\leq\; \pi_k + U_k$ where

$$U_k \;=\; \max_{\substack{i \in \mathcal{E}:\\ i \preceq k}} d_i + \sum_{\substack{a=(i,j) \in \mathcal{A}_{\text{nice}}:\\ j \preceq k}} (d_a - s_a)_+ + \sum_{\substack{(i,j) \in \mathcal{A}_{\text{head}}:\\ g_{ij}=0, \pi_i > \pi_j, j \preceq k}} \pi_i - \pi_j + L_{ij}. \tag{14}$$

We prove the claim by induction. For the first event, we have $\tilde{x}_1 = \pi_1 + d_1 \leq \pi_1 + U_1$. Now consider $\tilde{x}_k$. We distinguish the following three cases depending on which term in the definition (12) of $\tilde{x}$ is maximal:

– $\tilde{x}_k = \pi_k + d_k$. Since $d_k \leq U_k$, the claim is true.

– $\tilde{x}_k = \tilde{x}_i + L_a + d_a$ for $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}}$. We assume that $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$ or that $(i,k) \in \mathcal{A}_{\text{head}}$ with $\pi_i < \pi_k$. Then

$$\begin{aligned}\tilde{x}_k &= \tilde{x}_i + L_a + d_a \leq \pi_i + U_i + L_a + d_a\\ &\leq \underbrace{\pi_i + L_a}_{\leq \pi_k} + \max_{\substack{i' \in \mathcal{E}:\\ i' \preceq i}} d_{i'} + d_a + \sum_{\substack{a'=(i',j) \in \mathcal{A}_{\text{nice}}:\\ j \preceq i}} (d_{a'} - s_{a'})_+ + \sum_{\substack{(i',j) \in \mathcal{A}_{\text{head}}\\ g_{i'j}=0, \pi_{i'} > \pi_j, j \preceq i}} \pi_{i'} - \pi_j + L_{i'j}\\ &\leq \pi_k + U_k, \text{ hence the claim is true.}\end{aligned}$$

– $\tilde{x}_k = \tilde{x}_i + L_a + d_a$ for $(i,k) \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{fix}}$ where $(i,k) \in \mathcal{A}_{\text{head}}$ with $\pi_i > \pi_k$ and $g_{ik} = 0$. Then

$$\begin{aligned}\tilde{x}_k &= \tilde{x}_i + L_{ik} \leq \pi_i + U_i + L_{ik}\\ &\leq \max_{\substack{i' \in \mathcal{E}:\\ i' \preceq i}} d_{i'} + \sum_{\substack{a'=(i',j) \in \mathcal{A}_{\text{nice}}:\\ j \preceq i}} (d_{a'} - s_{a'})_+\\ &\quad + \pi_k + (\pi_i - \pi_k + L_{ik}) + \sum_{\substack{(i',j) \in \mathcal{A}_{\text{head}}\\ g_{i'j}=0, \pi_{i'} > \pi_j, j \preceq i}} \pi_{i'} - \pi_j + L_{i'j} d \leq \pi_k + U_k,\end{aligned}$$

and again the claim follows.

□

Using this result, we can give an upper bound on the minimal size needed for M:

Corollary 2. *$M \geq D$ is "large enough".*

Proof. We show that each timetable that satisfies (2)-(3), (6)-(9), (4) for all $a \in \mathcal{A}_{\text{change}}$ with $z_a = 0$ and (5) for all $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 0$ also satisfies (4) for all $a \in \mathcal{A}_{\text{change}}$ with $z_a = 1$ and (5) for all $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 1$ if $M \geq D$.

For each changing activity $a = (i,j) \in \mathcal{A}_{\text{change}}$, $\pi_j - \pi_i \geq L_a$, hence (using Theorem 1)

$$M \geq D \geq x_i - \pi_i \geq x_i - \pi_j + L_a \overset{(2)}{\geq} x_i - x_j + L_a,$$

so (4) indeed is satisfied for all $a \in \mathcal{A}_{\text{change}}$.

Now, let $(i,j) \in \mathcal{A}_{\text{head}}$ with $g_{ij} = 0$. We have to show that $M + x_i - x_j \geq L_{ji}$

Case 1: $\pi_i < \pi_j$. We use the proof of Theorem 1: For each j, each term that is added to U_j also is added to D. As $\pi_i < \pi_j$ and $g_{ij} = 0$, the term $\pi_j - \pi_i + L_{ji}$ is added to D, but not to U_j, hence $D - U_j \geq \pi_j - \pi_i + L_{ji}$ and we obtain

$$x_j \leq \pi_j + U_j \leq \pi_j + D - (\pi_j - \pi_i + L_{ji}) = D + \pi_i - L_{ji} \overset{(2)}{\leq} D + x_i - L_{ji}.$$

Case 2: $\pi_i > \pi_j$. As π is a feasible timetable, $\pi_i > \pi_j$ implies $\pi_i - \pi_j \geq L_{ji}$. Using Theorem 1,

$$M \geq D \geq x_j - \pi_j \geq x_j - \pi_i + L_{ji} \overset{(2)}{\geq} x_j - x_i + L_{ji}.$$

Both cases show that (5) is indeed satisfied for all $(i, j) \in \mathcal{A}_{\text{head}}$. □

3 Reducing the Complexity of the Integer Program

Headway constraints make the delay management problem hard to solve; due to headway constraints, a delay from a subsequent train might be carried over even to a train which has been scheduled earlier if the order of both trains is switched. At a first glance, it seems that all of the headway activities can carry over a delay to a previous train in an optimal solution. However, there indeed are some headway activities that can be neglected beforehand as we will show in this section. This reduction helps to solve (DM) more efficiently. We will introduce an algorithm to reduce the network in time $\mathcal{O}(|\mathcal{A}|)$ before solving the remaining NP-hard problem.

Definition 3. *Let $\mathcal{A}' \subseteq \mathcal{A}$. For $i \in \mathcal{E}$, we define the* successors *of i in $(\mathcal{E}, \mathcal{A}')$ and the* predecessors *of i in $(\mathcal{E}, \mathcal{A})$ as*

$$\begin{aligned} suc(i, \mathcal{A}') &:= \{j \in \mathcal{E} \setminus \{i\} : \text{ there exists a directed path from } i \text{ to } j \text{ in } (\mathcal{E}, \mathcal{A}')\}, \\ pre(i) &:= \{j \in \mathcal{E} \setminus \{i\} : \text{ there exists a directed path from } j \text{ to } i \text{ in } (\mathcal{E}, \mathcal{A})\}. \end{aligned}$$

Using this notation, we introduce the following algorithm:

```
Algorithm mark:
Input: The event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ and source delays $d_j > 0$ ($d_a > 0$) for some events $j \in \mathcal{E}$ (and for some activities $a = (i, j) \in \mathcal{A}_{\text{nice}}$, respectively).
Step 1: Set $\mathcal{A}_\pi := \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}} \cup \{(i, j) \in \mathcal{A}_{\text{head}} : \pi_i < \pi_j\}$.
Step 2: For each source delay $d_j > 0$, $j \in \mathcal{E}$ ($d_a > 0$, $a = (i, j) \in \mathcal{A}_{\text{nice}}$): mark $j$ and all $k \in suc(j, \mathcal{A}_\pi)$.
```

Note that we do not use all headway activities in the algorithm. The next theorem shows that this is in fact correct. In its proof and throughout this paper, we use $\mathcal{A}_\pi$ as defined in Step 1 of algorithm `mark`.

Theorem 4. *Let $L_a > 0$ $\forall a \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$, $L_{ij} > 0$ $\forall (i, j) \in \mathcal{A}_{\text{head}}$ and $w_i > 0$ $\forall i \in \mathcal{E}$. Let (x, z, g) be an optimal solution of (DM). Then, the following holds:*

$$i \in \mathcal{E} \text{ is not marked by algorithm } \texttt{mark} \;\Rightarrow\; x_i = \pi_i.$$

Proof. By contradiction. Assume $\exists i \in \mathcal{E}$ such that i does not get marked by algorithm `mark` and $x_i > \pi_i$. Assume that i is an event with minimal x among all such events.

First, by the construction of algorithm `mark`, we have $d_i = 0$, and for each $a = (k, i) \in \mathcal{A}_\pi$, $d_a = 0$ and k isn't marked as well (if i is not marked, also $pre(i)$ is not marked). As k is not marked and we assumed i to be the event with minimal x among all events j with $x_j > \pi_j$ that are not marked, we have $x_k = \pi_k$.

Now, we show that reducing x_i to $\tilde{x}_i := \pi_i < x_i$ also yields a feasible solution for (DM):

- As $d_i = 0$, $\tilde{x}_i = \pi_i$ satisfies (2).
- For all incoming activities $a = (k,i) \in \mathcal{A}_\pi$ we use $x_k = \pi_k$ and $d_a = 0$ to derive $\tilde{x}_i - x_k = \pi_i - \pi_k \geq L_a = L_a + d_a$, hence (3)-(5) hold for each $a = (k,i) \in \mathcal{A}_\pi$.
- For all outgoing activities $(i,k) \in \mathcal{A}$ we obtain $x_k - \tilde{x}_i > x_k - x_i \geq L_a + d_a$ where the last step holds since x is a feasible timetable. Consequently (3)-(5) hold for *all* $a = (i,k) \in \mathcal{A}$.
- Now let $(k,i) \in \mathcal{A}_{\text{head}} \setminus \mathcal{A}_\pi$. If $g_{ki} = 1$, (5) is satisfied due to Corollary 2, so assume $g_{ki} = 0$. We define $\tilde{g}_{ki} := 1$ and $\tilde{g}_{ik} := 0$; then

$$x_k - \tilde{x}_i = x_k - \pi_i \geq \pi_k - \pi_i \geq L_{ki}$$

since $(k,i) \notin \mathcal{A}_\pi$, i.e. $\pi_k > \pi_i$, so (5) holds for (i,k). Due to Corollary 2, (5) also holds for (k,i).

So $(\tilde{x}, z, \tilde{g})$ is a feasible solution with strictly better objective value than (x,z,g) (as $w_i > 0$), a contradiction to the optimality of (x,z,g). □

If we allow $w_i = 0$, then the following modification of Theorem 4 holds:

Theorem 5. *Let $L_a > 0$ $\forall a \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}_{\text{change}}$, $L_{ij} > 0$ $\forall (i,j) \in \mathcal{A}_{\text{head}}$ and $w_i \geq 0$ $\forall i \in \mathcal{E}$. Then there exists an optimal solution (x,z,g) of (DM) with:*

$$i \in \mathcal{E} \text{ is not marked by algorithm } \texttt{mark} \;\Rightarrow\; x_i = \pi_i.$$

Due to these results, we can use algorithm `mark` to reduce the size of the MIP formulation:

- Run algorithm `mark` on an instance of (DM).
- Delete events that are not marked (unless they have a source-delayed outgoing activity) and activities whose start or end event is not marked (unless they are source-delayed).
- Solve (DM) for this reduced instance.
- Set $x_i = \pi_i$ for all events $i \in \mathcal{E}$ that have been deleted in the second step.

Our numerical results show that reducing the network by algorithm `mark` as a preprocessing step significantly decreases the time needed to solve the IP, see page 8.

The results from Theorem 4 and Theorem 5 also can be used to tighten the upper bound on the minimal size needed for M from Corollary 2: It is sufficient to use the reduced network from above to calculate M.

There is another advantage of our results: Using algorithm `mark`, we can extend the *never-meet property* (see [Sch06,Sch07b]) to capacitated delay management problems, and show that – if this property holds for a given instance of (DM) – our objective (1) coincides with the sum of all delays that each passenger has at his or her final destination. The definition of this property is given next.

Definition 6. *An instance of (DM) has the* never-meet property *if*

- *for each source delay $d_j > 0$ with $j \in \mathcal{E}$ (or $d_a > 0$ with $a = (i,j) \in \mathcal{A}$), $suc(j, \mathcal{A}_\pi)$ is an out-tree, and if*
- *for each pair of different source delays $d_j > 0$ ($d_a > 0$, $a = (i,j)$) and $d_{\tilde{j}} > 0$ ($d_{\tilde{a}} > 0$, $\tilde{a} = (\tilde{i}, \tilde{j})$), we have $suc(j, \mathcal{A}_\pi) \cap suc(\tilde{j}, \mathcal{A}_\pi) = \emptyset$.*

This means that the never-meet property is satisfied if no event can be influenced (directly or indirectly) by more than one source delay. Note that one could define the never-meet property in any network (e.g. also in $\mathcal{N}$) but due to the circles of the headway constraints it would never be satisfied. Hence it is crucial to use the results of Theorem 4.

Lemma 7. *Given an instance of (DM) that satisfies the never-meet property and an optimal solution (x,z,g), assume that $z_a = 1$ for some $a = (i,j) \in \mathcal{A}_{\text{change}}$. Then, for each $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, k is not marked by algorithm* `mark`*, applied to $(\mathcal{E}, \mathcal{A} \setminus \{a\})$.*

Proof. By contradiction. Let $a = (i,j) \in \mathcal{A}_{\text{change}}$ and $z_a = 1$ in an optimal solution (x,z,g). Assume that there exists $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$ that is marked by algorithm `mark`, applied to $(\mathcal{E}, \mathcal{A} \setminus \{a\})$. Then, by construction of algorithm `mark`, there exists a directed path p_1 from $\tilde{j}_1$ to k in $(\mathcal{E}, \mathcal{A}_\pi \setminus \{a\})$ with either $d_{\tilde{j}_1} > 0$ or $d_{\tilde{a}_1} > 0$ for some $\tilde{a}_1 = (\tilde{i}_1, \tilde{j}_1)$.

As $z_a = 1$ and $w_a > 0$, there has to be a reason why a is not maintained, namely because of a delay of i. Hence, according to Theorem 4, i is marked by algorithm `mark`, so there exists a directed path p_2 from $\tilde{j}_2$ to i in $(\mathcal{E}, \mathcal{A}_\pi \setminus \{a\})$ with either $d_{\tilde{j}_2} > 0$ or $d_{\tilde{a}_2} > 0$ for some $\tilde{a}_2 = (\tilde{i}_2, \tilde{j}_2)$. As $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, p_2 can be extended to a path p_3 from $\tilde{j}_2$ to k in $(\mathcal{E}, \mathcal{A}_\pi)$ that contains a.

This is a contradiction to the never-meet property: either $\tilde{j}_1 = \tilde{j}_2$, then $suc(\tilde{j}_1, \mathcal{A}_\pi)$ is not an out-tree as we have two different paths p_1 (not containing a) and p_3 (containing a) from $\tilde{j}_1$ to k, or $\tilde{j}_1 \neq \tilde{j}_2$, then $suc(\tilde{j}_1, \mathcal{A}_\pi) \cap suc(\tilde{j}_2, \mathcal{A}_\pi) \supseteq \{k\} \neq \emptyset$. □

Corollary 8. *Given an instance of (DM) that satisfies the never-meet property and an optimal solution (x,z,g), assume that $z_a = 1$ for some $a = (i,j) \in \mathcal{A}_{\text{change}}$. Then, for each $k \in suc(j, \mathcal{A}_\pi) \cup \{j\}$, $x_k = \pi_k$.*

Proof. If, in an optimal solution, $z_a = 1$ for some $a \in \mathcal{A}_{\text{change}}$, this solution is also an optimal solution for the same instance with event-activity network $(\mathcal{E}, \mathcal{A} \setminus \{a\})$. From Lemma 7 we know that k is not marked for all $k \in suc(j, \mathcal{A}_\pi)$. Theorem 4 hence completes the proof. □

Theorem 9. *If the never-meet property holds, (DM) is equivalent to minimizing the sum of all delays of all passengers at their final destinations.*

Proof. The proof can be done analogously to the result in [Sch07b] since the ingredient needed is provided in Corollary 8. □

Numerical Results

To test the efficiency of our reduction, we used railway data from the Harz region in the center of Germany (in form of a periodic timetable, including headway and turnover activities), originally used in [LSS+07]. We consider all passenger railway lines within this region as well as 9 long-distance lines. The dataset contains 598 stations, 92 trains (vehicles) and 30 lines, each line with two directions. We take into account all events and all activities that take place in an 8-hours time window. Details on how the periodic timetable is expanded to an aperiodic event-activity network are described in [LSS+07]. See Figure 1 for a sketch of the part of the German railway network which we consider.

The resulting event-activity network contains 21 269 events and 39 985 activities. We generated 1 000 different delay scenarios. In each delay scenario, 25 randomly chosen driving or waiting activities have been delayed by a random delay between 60 and 1 200 seconds. In both cases, (DM) was solved using Xpress-MP 2007B on an AMD Opteron 275 system with 4 GB RAM.

The results clearly show the benefit of a preprocessing step based on algorithm `mark`: In 13 of 1 000 test cases, the original problem could not be solved due to an "out of memory" error – in all those cases, we got an optimal solution for the reduced problem. In average, the time needed to solve the reduced problems was only 19.1% of the time needed to solve the original problems. Especially, in 929 cases, the computation time was reduced to less than 50%, and in 332 cases (nearly one third of all cases), it was even reduced to less than 10%.

4 Heuristics

Although the results from Theorem 4 and Theorem 5 can be used for introducing a preprocessing step that speeds up the computation of an optimal solution, very large instances of (DM) cannot be solved exactly in a reasonable amount of time. To be able to provide at least some solution

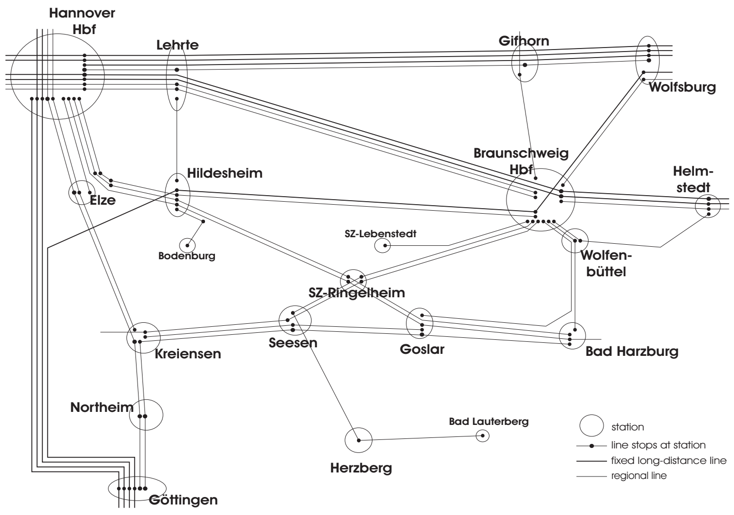

Fig. 1. The part of the German railway network which is considered for our numerical results. (The Figure is taken from [LSS+07].)

for these cases, we introduce heuristics that can solve even large instances by approximation. The idea of our heuristics is to fix the order of trains (i.e. the variables g_{ij}) in advance and solve the remaining uncapacitated delay management problem (UDM). Such a fixing can be done in many ways; four of them will be discussed next. In the first approach we fix the order of the trains according to the original timetable.

Algorithm `First scheduled, first served` **(FSFS):**

1. Set $\mathcal{A}_{\text{fix}} := \{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i \leq \pi_j\}$.
2. Compute the exact solution of (UDM($\mathcal{A}_{\text{fix}}$)).

In the next heuristic, we first neglect all capacity constraints and solve the uncapacitated delay management problem (UDM). We then fix the order of the trains according to the optimal disposition timetable x of (UDM).

Algorithm `First rescheduled, first served` **(FRFS):**

1. Solve the corresponding problem (UDM) and obtain disposition timetable x.
2. Set $\mathcal{A}_{\text{fix}} := \{(i,j) \in \mathcal{A}_{\text{head}} : x_i \leq x_j\}$.
3. Compute the exact solution of the corresponding instance of (UDM($\mathcal{A}_{\text{fix}}$)).

Solving an additional instance of (UDM) in the first step increases the running time of FRFS compared to FSFS. In the third heuristic, we again start by solving the corresponding instance of (UDM) and fix the order of trains according to an optimal disposition timetable, but here we additionally fix the wait-depart decisions obtained from the solution of (UDM) such that we can compute a feasible disposition timetable for (DM) by applying the critical path method. In contrast to FRFS, Step 3 can now be solved very efficiently.

Algorithm `FRFS with early connection fixing` **(EARLYFIX):**

1. Solve the corresponding problem (UDM) and obtain a disposition timetable x and values for z.
2. Set $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : x_i \le x_j\}$.
3. Compute the solution of $(\text{PP}(\mathcal{A}_{\text{fix}}))$.

Since the first step in both FRFS and in EARLYFIX is to solve (UDM), we obtain from these heuristics not only an approximation of the optimal solution, but also a lower bound on its objective value, and hence their absolute errors can be bounded a posteriori by

$$F^{\text{FRFS}} - F^{\text{UDM}} \text{ and } F^{\text{EARLYFIX}} - F^{\text{UDM}},$$

respectively, where F^{FRFS} and F^{EARLYFIX} are the objective values of FRFS and EARLYFIX. Comparing these heuristics we obtain the following result.

Lemma 10. $F^{\text{DM}} \le F^{\text{FRFS}} \le F^{\text{EARLYFIX}}$.

The last heuristic we tested is the only one with polynomial runtime. It is a modification of the FSFS heuristic we presented first. As we do in FSFS, we fix the order of trains according to the original timetable π, but instead of solving the remaining problem exactly, we use a heuristic approach to fix the wait-depart decisions. The idea is to maintain the "most important" connections and do not care about the less important ones.

Algorithm `FSFS with priority-based fixing` **(PRIORITY):**

1. Maintain the "most important" connections:
   - Sort the changing edges in descending order according to their weights w_a.
   - Set $z_a = 0$ for the first $k\%$ of the connections.
2. Set $\mathcal{A}_{\text{fix}} := \{a \in \mathcal{A}_{\text{change}} : z_a = 0\} \cup \{(i,j) \in \mathcal{A}_{\text{head}} : \pi_i \le \pi_j\}$.
3. Compute the exact solution of $(\text{PP}(\mathcal{A}_{\text{fix}}))$.

Comparing PRIORITY to FSFS, we obtain the following relation.

Lemma 11. *Let F^{FSFS} and F^{PRIORITY} denote the objective values of the solutions computed by FSFS and PRIORITY, respectively. Then $F^{\text{DM}} \le F^{\text{FSFS}} \le F^{\text{PRIORITY}}$*

Finding bounds on the relative error of these heuristics is in general not possible: the results of all heuristics might get arbitrarily bad compared to the optimal solution. In our first result we prove that fixing the priority decisions according to the original timetable might become arbitrarily bad, while Lemma 13 shows that fixing the priority decisions according to the optimal solution of the corresponding (UDM) might also become arbitrarily bad. Hence, both groups of heuristics FSFS and PRIORITY as well as FRFS and EARLYFIX can become arbitrarily bad. However, we are able to bound the relative error of EARLYFIX using the input data of the special instance in Theorem 15.

Lemma 12. *Let* HEU *be a heuristic that solves (DM), fixing the g_{ij} variables as they are set in the original timetable, and let F^{HEU} its objective value. Then for each $k \in \mathbb{N}$, there exists an instance of (DM) such that*

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} > k.$$

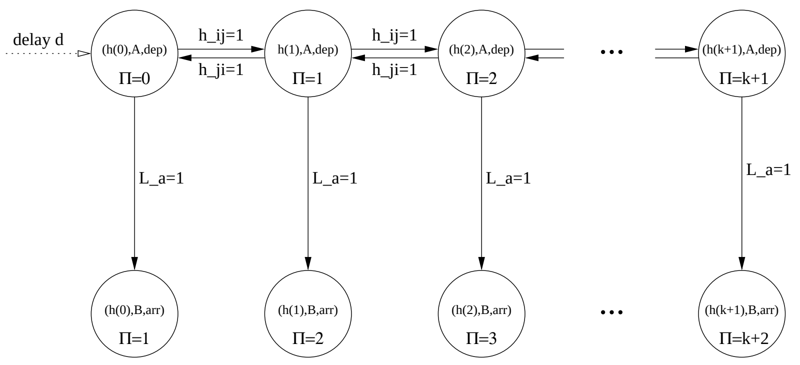

Fig. 2. Event-activity network for the proof of Lemma 12.

Proof. Let $k \in \mathbb{N}$. Assume that we have two stations A and B and $k+2$ trains $h_0, h_1, \ldots, h_{k+1}$. All trains drive from station A to station B. By (t, s, arr) and (t, s, dep), we denote the arrival of train t at station s and its departure from that station. $\pi(t, s, \text{arr})$ denotes the time of such an event in the original timetable. In the original timetable, the trains in our instance leave station A in the order $h_0, h_1, \ldots, h_{k+1}$ at the times $\pi(h_i, A, \text{dep}) = i$ and arrive at station B at the times $\pi(h_i, B, \text{arr}) = i+1$, $i \in \{0, \ldots, k+1\}$. For each $i \in \{0, \ldots, k\}$, the departure of train h_i and the departure of train h_{i+1} are connected by a pair of headway edges. All weights and all lower bounds are set to 1. The resulting event-activity network is shown in Figure 2.

Now, assume that (h_0, A, dep) is delayed by $d \geq k+2$. In the optimal solution, the trains $h_1, \ldots, h_{k+1}$ leave and arrive on time, while train h_0 has a delay of d, so $F^{\text{DM}} = d$. If we solve the problem by a heuristic that sets the g_{ij} variables as they are set in the original timetable without delays, all trains get a delay of at least d, so $F^{\text{HEU}} \geq (k+2) \cdot d$, hence

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} \geq \frac{(k+2) \cdot d - d}{d} = k+1 > k.$$

□

Similarly, one can show the following result concerning FSFS and PRIORITY.

Lemma 13. *Let* HEU *be a heuristic that solves (DM), fixing the g_{ij} variables as they are set in the optimal solution of the corresponding problem (UDM). Then for each $k \in \mathbb{N}$, there exists an instance of (DM) such that*

$$\frac{F^{\text{HEU}} - F^{\text{DM}}}{F^{\text{DM}}} > k.$$

However – as we will show at then end of this Section – the heuristics do not behave as bad as one might think regarding the results above. Moreover, using the input data of the specific instance we might be able to derive upper bounds. This is exemplarily done to bound the relative error of EARLYFIX.

Lemma 14. *Let x^{relax} be an optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) with events $\mathcal{E}$ and activities $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{nice}} \cup \mathcal{A}^{\text{fix}}_{\text{change}}$ and x^{cap} an optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) with events $\mathcal{E}$ and activities $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{nice}} \cup \mathcal{A}^{\text{fix}}_{\text{change}} \cup \mathcal{A}^{\text{fix}}_{\text{head}}$. Let $\mathcal{A}^1 := \mathcal{A}^{\text{fix}}_{\text{head}} \cap \mathcal{A}_\pi$ and $\mathcal{A}^2 := \mathcal{A}^{\text{fix}}_{\text{head}} \setminus \mathcal{A}_\pi$. Then, we have*

$$x_i^{\text{cap}} \leq x_i^{\text{relax}} + \sum_{\substack{(k,l)\in\mathcal{A}^1:\\ k\in pre(i)}} (x_k^{\text{relax}} - \pi_k) + \sum_{\substack{(k,l)\in\mathcal{A}^2\\ k\in pre(i)}} (x_k^{\text{relax}} + L_{kl}) \quad \forall i \in \mathcal{E}. \tag{15}$$

Proof. An optimal solution of (PP($\mathcal{A}_{\text{fix}}$)) can be computed by applying the critical path method that has been introduced in Section 2. We prove (15) by induction and distinguish two cases depending which term in (12) gets maximal. For $k = 1$ and for $x_k^{\text{cap}} = \pi_k + d_k$, (15) is true. We therefore assume $x_k^{\text{cap}} > \pi_k + d_k$ and that (15) is true for all $j < k$. Let

$$\tilde{a} = (j,k) \;:=\; \underset{a=(j,k)\in\mathcal{A}_{\text{fix}}}{\text{argmax}} \; x_j^{\text{cap}} + L_a + d_a.$$

Case 1: Assume that $\tilde{a} \in \mathcal{A}_{\text{nice}} \cup \mathcal{A}^{\text{fix}}_{\text{change}}$. Then, $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{\tilde{a}} + d_{\tilde{a}}$, and using $x_k^{\text{relax}} - x_j^{\text{relax}} \geq L_{\tilde{a}} + d_{\tilde{a}}$, (15) to estimate x_j^{cap} and $pre(j) \subset pre(k)$, we see that (15) is satisfied.

Case 2: Assume that $\tilde{a} \in \mathcal{A}^1$. Then, $d_{\tilde{a}} = 0$ and $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{jk}$. Using (15), we get

$$x_k^{\text{cap}} \;\leq\; x_k^{\text{relax}} + \sum_{\substack{(l,m)\in\mathcal{A}^1:\\ l\in pre(j)}} (x_l^{\text{relax}} - \pi_l) + \sum_{\substack{(l,m)\in\mathcal{A}^2:\\ l\in pre(j)}} (x_l^{\text{relax}} + L_{lm}) + x_j^{\text{relax}} - x_k^{\text{relax}} + L_{jk}. \tag{16}$$

Using $\pi_k - \pi_j \geq L_{jk}$, $x_k^{\text{relax}} \geq \pi_k$ and $pre(j) \subset pre(k)$, we see that (15) holds.

Case 3: Assume that $\tilde{a} \in \mathcal{A}^2$. Then, $d_{\tilde{a}} = 0$ and $x_k^{\text{cap}} = x_j^{\text{cap}} + L_{jk}$. As in the second case, we get inequality (16). We use $x_k^{\text{relax}} \geq 0$, move $x_j^{\text{relax}} + L_{jk}$ to the second sum and use $pre(j) \subset pre(k)$ to prove the lemma for the third case. □

We can use Lemma 14 to get an upper bound on the relative error of EARLYFIX: We replace x^{relax} by x^{DM} and x^{cap} by x^{EARLYFIX}, and define $\mathcal{A}^{\text{fix}}_{\text{head}} = \{(i,j) \in \mathcal{A}_{\text{head}} : x_i^{\text{DM}} \leq x_j^{\text{DM}}\}$. Using the delay $y_i = x_i - \pi_i$ of event i instead of its time x_i in the disposition timetable, we have

$$y_i^{\text{EARLYFIX}} - y_i^{\text{DM}} \;\leq\; \sum_{\substack{(k,l)\in\mathcal{A}^1:\\ k\in pre(i)}} y_k^{\text{DM}} + \sum_{\substack{(k,l)\in\mathcal{A}^2:\\ k\in pre(i)}} (y_k^{\text{DM}} + \pi_k + L_{kl}) \;\leq\; F^{\text{UDM}} + \sum_{(k,l)\in\mathcal{A}^{\text{fix}}_{\text{head}}} (L_{kl} + \pi_k),$$

where the second inequality holds if we assume $w_i \geq 1 \forall\; i \in \mathcal{E}$. With this assumption, we hence obtain

$$F^{\text{EARLYFIX}} - F^{\text{DM}} \;\leq\; \left(F^{\text{UDM}} + \sum_{(k,l)\in\mathcal{A}^{\text{fix}}_{\text{head}}} (L_{kl} + \pi_k) \right) \sum_{i\in\mathcal{E}} w_i.$$

If, in addition, $\mathcal{A}^2 = \emptyset$, we have $F^{\text{EARLYFIX}} - F^{\text{DM}} \;\leq\; F^{\text{UDM}} \sum_{i\in\mathcal{E}} w_i$. This yields the following result.

Theorem 15. *Consider an instance of (DM) with weights $w_i \geq 1$ for all $i \in \mathcal{E}$.*

a) If the solution x^{DM} of (UDM) satisfies $\pi_i \leq \pi_j \;\Rightarrow\; x_i^{\text{DM}} \;\leq\; x_j^{\text{DM}} \quad \forall\, (i,j) \in \mathcal{A}_{\text{head}}$, then

$$\frac{F^{\text{EARLYFIX}} - F^{\text{DM}}}{F^{\text{DM}}} \;\leq\; \sum_{i\in\mathcal{E}} w_i.$$

b) If $F^{\text{DM}} \geq 1$, we have

$$\frac{F^{\text{EARLYFIX}} - F^{\text{DM}}}{F^{\text{DM}}} \;\leq\; \left(1 + \sum_{(k,l)\in\mathcal{A}^1} (L_{kl} + \pi_k) \right) \sum_{i\in\mathcal{E}} w_i.$$

The theorem gives rise to the assumption that the quality of the solution depends on the size of $\sum_{(k,l)\in\mathcal{A}^1} L_{kl}$.

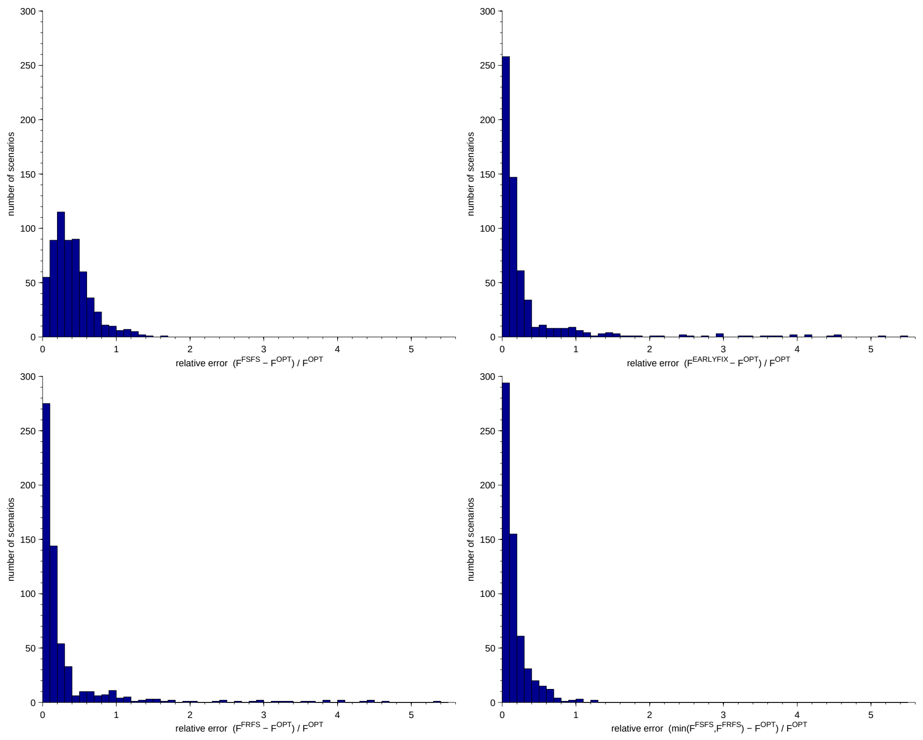

Fig. 3. The relative error of FSFS, EARLYFIX, FRFS and of the combination of FSFS and FRFS (observation period of 3 hours, 7 104 events, 9 570 activities).

Numerical Results

The dataset on which our numerical results for the heuristics are based is the same dataset as in Section 3. The sizes of the event-activity network for different observation periods are stated in Table 2. The IP formulation was solved using Xpress-MP 2006 on a Pentium IV 3 GHz processor with 2 GB RAM. We generated about 600 different delay scenarios; in each of them, we assigned 25 randomly generated source delays of 1-20 minutes to 25 randomly chosen driving and waiting activities. We also know the weights $\tilde{w}_a$ of the changing activities $a \in \mathcal{A}_{\text{change}}$. We hence set $w_i = 1$ for all events $i \in \mathcal{E}$ and $w_a = \frac{\tilde{w}_a}{\bar{w}}$ for all $a \in \mathcal{A}_{\text{change}}$. $\bar{w}$ is the arithmetic mean of the weights $\tilde{w}_a$ which is used in order to prevent an overestimation of the missed connections.

In Figure 3, we present four histograms of the relative errors for the heuristics FSFS, EARLYFIX and FRFS and for the approach running both FSFS and FRFS and taking the better solution. We took into account all events and all activities that take place during a fixed observation period of 3 hours. On the x-axis we graphed intervals of 0.1 length describing the relative error. The first interval corresponds to a relative error between 0 and 0.1, the second interval to a relative error between 0.1 and 0.2, and so on. We show in how many of the about 600 different delay scenarios the relative error of the respective heuristic takes a value in an interval of length 0.1 – for example, in about 55 scenarios out of 600, the relative error of FSFS is in the interval [0, 0.1].

FRFS is slightly better than EARLYFIX concerning the quality of their solutions. For both of them, the number of scenarios with a small relative error is significantly higher than for FSFS. On the other hand, there are some scenarios in which EARLYFIX and FRFS have a very high relative error – FSFS does not have these outliers. If we combine FSFS and FRFS – this means that for each scenario, we take the solution with the smaller objective value – we benefit from the large number of scenarios with a small relative error in FRFS and from the fact that FSFS does not have outliers as FRFS does have.

	observation period of		
heuristic	3 hours	6 hours	10 hours
FSFS	141 (23.58%)	239 (39.97%)	263 (43.98%)
EARLYFIX	219 (36.62%)	83 (13.88%)	75 (12.54%)
FRFS	457 (76.42%)	361 (60.37%)	336 (56.19%)

Table 1. How often (out of 598 different scenarios) is FSFS, EARLYFIX and FRFS at least as good as the two other heuristics, w.r.t different observation periods?

Table 1 shows the quality of FSFS, EARLYFIX and FRFS compared to each other. We specify for each heuristic in how many cases it computes a solution at least as good as the solutions of the other heuristics. We take into account different observation periods. For larger event-activity networks, EARLYFIX performs quite bad, while the number of scenarios in which FSFS computes the best solution grows significantly.

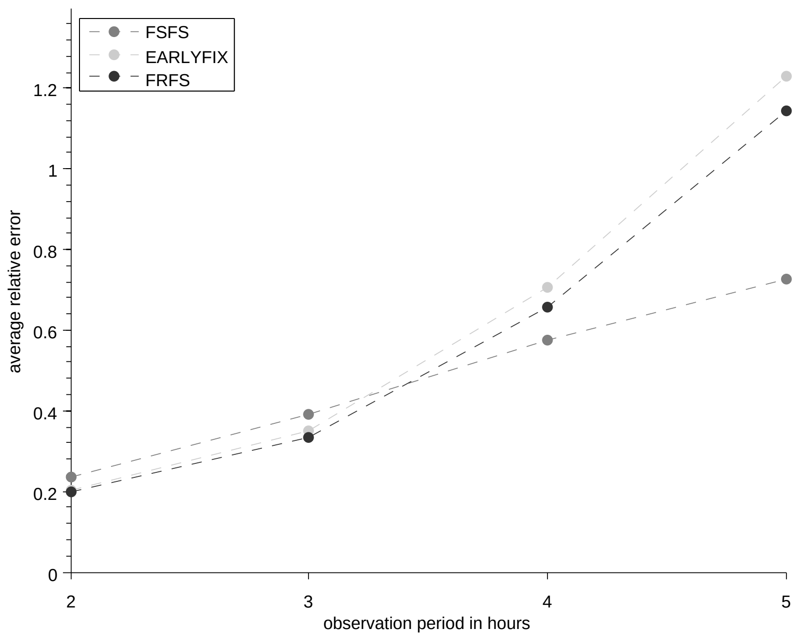

Fig. 4. Average relative error of FSFS, EARLYFIX and FRFS for different observation periods between two and five hours.

In Figure 4, we show how the relative errors of FSFS, EARLYFIX and FRFS grow with the length of the observation period, i.e. with the size of the relevant event-activity network. The larger the event-activity network, the larger the relative error of all heuristics.

In Table 2, we finally specify the runtime of the exact solution and of the heuristics FSFS, FRFS and EARLYFIX for different observation periods (i.e. for different sizes of the event-activity network). An observation period of k hours means that we considered events and activities during a fixed k-hours time slot. It turns out that all heuristics are significantly faster than the optimal solution (calculated via the ILP formulation by Xpress). EARLYFIX clearly outperforms FSFS and FRFS by a factor of 3. FSFS and FRFS are nearly equal considering the computation time.

size of the event-activity network					runtime (in seconds) of algorithm			
hours	$\vert\mathcal{E}\vert$	$\vert\mathcal{A}\vert$	$\vert\mathcal{A}_{\text{head}}\vert$	$\vert\mathcal{A}_{\text{change}}\vert$	exact	FSFS	EARLYFIX	FRFS
2	4 726	5 865	1 110	125	19.45	0.24	0.11	0.26
3	7 104	9 570	2 378	187	185.33	0.46	0.17	0.49
4	9 446	14 079	4 428	307	584.66	0.69	0.25	0.74
5	11 824	18 605	6 514	369	1 075.52	0.91	0.31	1.00
6	14 166	23 396	8 846	489	-	1.16	0.39	1.26
8	18 888	32 673	13 260	632	-	1.65	0.53	1.83
10	23 596	41 992	17 656	852	-	2.04	0.68	2.34
15	33 718	61 944	27 138	1 209	-	3.01	1.01	3.63

Table 2. Average runtime for different sizes of the event-activity network.

5 Conclusion and Further Research

In this paper, we presented and analyzed an integer programming formulation for the capacitated delay management problem. We suggest a reduction technique and four different heuristics. In our analysis it turns out that the headway activities which are not in the same direction as in the original timetable play a basic role. We were also able to give a reasonable definition of the never-meet property and to extend properties known from the uncapacitated delay management problem.

There are more properties of the never-meet property that are currently exploited, e.g. to extend the linear-time algorithm of the uncapacitated problem to the capacitated case. We also work on a deeper analysis of the heuristics and on new approaches such as machine-based learning techniques. Moreover, other issues should be considered to make the approach applicable in practice. Sometimes a change of the vehicle routes is appropriate to reduce delays, and often it is necessary to include the microscopic routes of the trains, in particular at large stations.

Acknowledgment. We want to thank Jens Dupont of *Deutsche Bahn* and Christian Liebchen of TU Berlin for providing the data for the case study.

References

[BGJ+05] N. Bissantz, S. Güttler, J. Jacobs, S. Kurby, T. Schaer, A. Schöbel, and S. Scholl. DisKon - Disposition und Konfliktlösungs-management für die beste Bahn. *Eisenbahntechnische Rundschau (ETR)*, 45(12):809–821, 2005. (in German).

[BHLS07] Andre Berger, Ralf Hoffmann, Ulf Lorenz, and Sebastian Stiller. Online delay management: Pspace hardness and simulation. Technical Report ARRIVAL-TR-0097, ARRIVAL Project, 2007.

[CS07] C. Conte and A. Schöbel. Identifying dependencies among delays. In *proceedings of IAROR 2007*, 2007. ISBN 978-90-78271-02-4.

[Elm77] S.E. Elmaghraby. *Activity Networks.* Wiley Interscience Publication, 1977.

[Gat07] M. Gatto. *On the Impact of Uncertainty on Some Optimization Problems: Combinatorial Aspects of Delay Management and Robust Online Scheduling.* PhD thesis, ETH Zürich, 2007.

[GGJ+04] M. Gatto, B. Glaus, R. Jacob, L. Peeters, and P. Widmayer. Railway delay management: Exploring its algorithmic complexity. In *Proc. 9th Scand. Workshop on Algorithm Theory (SWAT)*, volume 3111 of *LNCS*, pages 199–211, 2004.

[GHL08] L. Giovanni, G. Heilporn, and M. Labbé. Optimization models for the delay management problem in public transportation. *European Journal of Operational Research*, 189(3):762–774, September 2008. to appear.

[GJPS05] M. Gatto, R. Jacob, L. Peeters, and A. Schöbel. The computational complexity of delay management. In D. Kratsch, editor, *Graph-Theoretic Concepts in Computer Science: 31st International Workshop (WG 2005)*, volume 3787 of *Lecture Notes in Computer Science*, 2005.

[GJPW07] M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. On-line delay management on a single train line. In *Algorithmic Methods for Railway Optimization*, Lecture Notes in Computer Science. Springer, 2007. to appear.

[Gov98] R.M.P. Goverde. The max-plus algebra approach to railway timetable design. In *Computers in Railways VI: Proceedings of the 6th international conference on computer aided design, manufacture and operations in the railway and other advanced mass transit systems, Lisbon, 1998*, pages 339–350, 1998.

[GS07] A. Ginkel and A. Schöbel. To wait or not to wait? The bicriteria delay management problem in public transportation. *Transportation Science*, 41(4):527–538, 2007.

[LSS$^+$07] C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller, and A. Prigge. Computing delay-resistant railway timetables. Technical Report TR-0066, ARRIVAL Report, 2007. see http://arrival.cti.gr/index.php/Documents/Main.

[Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables.* Deutsches Zentrum für Luft– und Raumfahrt, Institut für Flugführung, Braunschweig, 1998. Habilitationsschrift.

[RdVM98] B. De Schutter R. de Vries and B. De Moor. On max-algebraic models for transportation networks. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 457–462, Cagliari, Italy, 1998.

[Sch01] A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

[Sch06] A. Schöbel. *Customer-oriented optimization in public transportation.* Optimization and Its Applications. Springer, New York, 2006.

[Sch07a] A. Schöbel. Capacity constraints in delay management. 2007. ARRIVAL Report TR-0017.

[Sch07b] A. Schöbel. Integer programming approaches for solving the delay management problem. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes in Computer Science, pages 145–170. Springer, 2007.

[SM99] L. Suhl and T. Mellouli. Requirements for, and design of, an operations control system for railways. In *Computer-Aided Transit Scheduling.* Springer, 1999.

[SMBG01] L. Suhl, T. Mellouli, C. Biederbick, and J. Goecke. Managing and preventing delays in railway traffic by simulation and optimization. In *Mathematical methods on Optimization in Transportation Systems*, pages 3–16. Kluwer, 2001.