Delay Management including Capacities of Stations

Twan Dollevoet[1,2] Dennis Huisman[1,2] Marie Schmidt[3]
Anita Schöbel[3]

[1] Econometric Institute and ECOPT, Erasmus University Rotterdam
P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands.
{dollevoet,huisman}@ese.eur.nl

[2] Process quality & Innovation, Netherlands Railways
P.O. Box 2025, NL-3500 HA Utrecht, the Netherlands.

[3] Institute for Numerical and Applied Mathematics, Georg-August University
Lotzestr. 16 - 18, D-37083 Göttingen, Germany.
{m.schmidt,schoebel}@math.uni-goettingen.de

Econometric Institute Report 2012-22

September 11, 2012

Abstract

The question of delay management is whether trains should wait for delayed feeder trains or should depart on time. Solutions to this problem strongly depend on the available capacity of the railway infrastructure. While the limited capacity of the tracks has been considered in delay management models, the limited capacity of the stations has been neglected so far. In this paper, we develop a model for the delay management problem that includes the stations' capacities. This model allows to reschedule the platform assignment dynamically. Furthermore, we propose an iterative algorithm in which we first solve the delay management model with a fixed platform assignment and then improve this platform assignment in each step. We show that the latter problem can be solved in polynomial time by presenting a totally unimodular IP formulation. Finally, we present an extension of the model that balances the delay of the passengers on the one hand and the number of changes in the platform assignment on the other. All models are evaluated on real-world instances from Netherlands Railways.

Keywords: Delay management, station capacities, platform assignment, integer programming, graph coloring

1 Introduction and Motivation

Since the first integer programming formulation for delay management in 2001 (Schöbel, 2001), there has been a significant amount of research on extensions of the basic delay management problem. Delay management deals with the question whether a train should better wait for a delayed feeder train or depart on time (*wait-depart decisions*). The goal is to

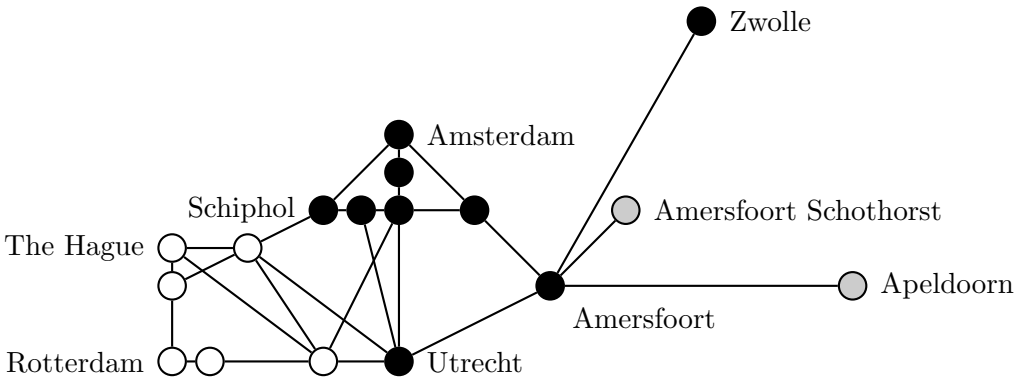

Figure 1: Part of the railway network in the Netherlands.

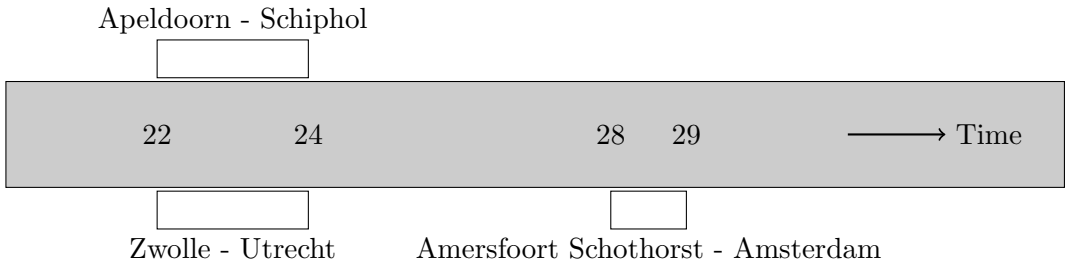

Figure 2: A schematic representation of the assignment of trains to platform tracks at Amersfoort

minimize total (weighted) passenger delay. Because missing a connection is an enormous frustration for railway passengers, delay management has also received much attention from railway companies. Unfortunately, the models so far do not include station capacities, which is a crucial aspect in practice, because station capacities are often limited. The topic of this paper is delay management with station capacities.

In most European countries, railway transport plays an important role. Many people travel by train for distances between 20 and 800 kilometers, especially during peak hours when there are many traffic jams on the highways. Passengers prefer a direct connection, however it is impossible that there is a direct connection between all possible origin-destination pairs. The Dutch line plan is constructed in such a way that most passengers (about 75%) have a direct connection. In addition, the most important transfers are cross-platform and have a short connection time. A typical example is station Amersfoort, where trains from the North and East arrive at the same time, and continue towards the directions of Utrecht and Schiphol Airport. Passengers in the train from Zwolle towards Utrecht (see Figure 1) that have Amsterdam Central Station as their final destination, can change on the same platform, where their train to Amsterdam departs a few minutes later.

We use this example to illustrate why station capacities should be taken into account in a delay management model. In the regular timetable the trains from Zwolle to Utrecht and from Apeldoorn to Schiphol Airport depart and arrive on the left and right side of the same platform in Amersfoort (see Figure 2) at minute .22 and .24, respectively. The train from

Amersfoort Schothorst towards Amsterdam arrives on the left side of this platform at minute .28 and leaves at .29. Now suppose that the train from Zwolle has a delay of 10 minutes and that there are many transferring passengers towards Amsterdam.

- The optimal solution of a basic delay management model suggests that the train towards Amsterdam waits for the transferring passengers from Zwolle. In addition, it assumes that this train arrives on time in Amersfoort. However, if this train would arrive on time and waits for the connecting train from Zwolle, this feeder train could never arrive, because the platform track is blocked.

- A possible strategy to obtain a feasible solution is to use the delay management model only to obtain the wait-depart decisions, but take the regular order of the trains. In that case, the train from Zwolle arrives at .32 with 10 minutes delay and then departs at .34. Then, after this train has departed, the train towards Amsterdam arrives at the same platform track. As a minimum headway time of 3 minutes is required between two trains using the same platform track, this means that this train will never arrive before minute .37. As a consequence, the train to Amsterdam will depart with a large delay.

- However, the right side of the platform is already empty for some time. If the train towards Amsterdam is rescheduled to this platform track, it can wait for the transferring passengers there. As the train from Zwolle arrives at .32 and two minutes of minimal transfer time are required, the train towards Amsterdam can leave at .34. This solution gives the minimum possible delay for the situation.

Of course, the real-time rescheduling of the platform assignment contains also some disadvantages. It requires additional work for dispatchers of the traffic control centers, and it is annoying for the passengers, especially, if they have to move to another platform.

From this example, we can draw the following conclusions.

- The optimal solution value of basic delay management models provides a lower bound on the optimal solution value of delay management with station capacities. However, this solution can be infeasible in practice.

- Fixing the wait-depart decisions of an optimal delay management solution and fixing the order of the trains leads to feasible solutions. These solutions are typically of low quality: Passengers face very large delays in these solutions.

- When re-assigning platform tracks is allowed, this may result in less passenger delays.

In this paper, we incorporate station capacities in the delay management model. We compare solutions with a fixed platform assignment to solutions in which we can re-assign a platform track during the operations. The contributions of this paper are as follows. Firstly, we present a new integer programming formulation for the delay management problem taking into account station capacity constraints. Secondly, we develop an iterative approach to solve this problem heuristically. Thirdly, we compare the optimal solution of the new model and the solution of the iterative heuristic with methods based on the traditional delay management model. Finally, we investigate the effect of flexibility in the platform assignment on the total passenger delay in several real-world problem instances of Netherlands Railways. Based on

our findings, railway companies can make a trade-off between changing platform tracks at the last moment versus the total passenger delay.

The remainder of the paper is structured as follows. Section 2 reviews the relevant literature. In Section 3 we present an integer programming formulation for the delay management model with station capacities. In Section 4 we discuss an iterative approach to solve this model. Computational results are discussed in Section 5. In Section 6, we discuss the balance between the passenger delay on the one hand and the number of platform track changes on the other. Finally, we finish the paper with some concluding remarks and suggestions for further research in Section 7.

2 Literature Review

There exist various models and solution approaches for delay management. 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. A first integer programming formulation for this problem has been given in Schöbel (2001) and has been further developed by De Giovanni et al. (2008) and Schöbel (2007); see also Schöbel (2006) for an overview about various models. The complexity of the problem has been investigated in Gatto et al. (2005). An online version of the problem has been studied by Gatto et al. (2007), Gatto (2007), Kliewer and Suhl (2011), and Krumke et al. (2011). Berger et al. (2011) show that it is PSPACE-hard. All models mentioned so far assume that passengers have to wait a complete cycle time in case they miss their connection. In order to compute the delay for the passengers more accurately, Dollevoet et al. (2012b) reroute passengers that have missed a connection. In order to solve large-case real-world instances, Dollevoet and Huisman (2011) present several heuristics for this delay management model with passenger rerouting.

In railway transportation, an important issue concerns the limited capacity of the track system. Schöbel (2009) presents a first model for delay management that includes capacity constraints. Schachtebeck and Schöbel (2010) and Schachtebeck (2010) give an integer programming formulation and propose heuristic methods for the capacitated delay management problem. The idea is to add *headway constraints*, which make sure that there is enough distance between two train departures and hence prevent two trains from using the same piece of track at the same time. Using machine scheduling models, it turns out that the model with headway constraints is NP-hard even in the case that no wait-depart decisions have to be made, see Conte and Schöbel (2007).

Another line of research dealing with railway operations is based on the alternative graph formulation (Mascis and Pacciarelli, 2002), originally used to model job shop variants. A branch-and-bound algorithm for finding a conflict-free train schedule, minimizing the largest delay, is developed in D'Ariano et al. (2007) and Caimi et al. (2011). In Corman et al. (2010), the authors suggest a tabu search to solve both the train sequencing and train routing problem, where a set of possible routes is given as input. The alternative graph formulation has been used in Corman et al. (2012) in the context of delay management, in a bi-objective approach that optimizes the maximal delay and number of missed connection simultaneously. A similar bi-objective approach is presented by Ginkel and Schöbel (2007) for the macroscopic model. Here, the total train delay and the number of missed connections are minimized.

Dollevoet et al. (2012a) made a first step towards a complete integration of delay management and train scheduling. They propose an optimization framework that iteratively solves first a

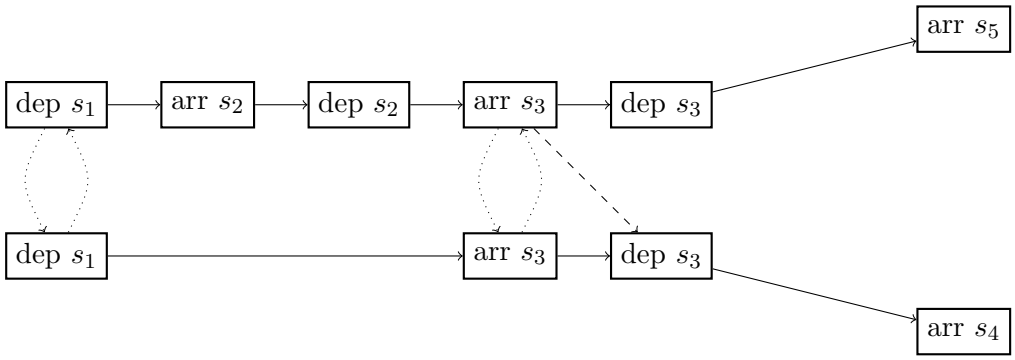

Figure 3: An event-activity network with a transfer activity and headway activities

delay management model and then a train scheduling model. In a computational study, it is shown that the approach converges quickly if the delays are small. For larger delays, the behavior of this approach is very unstable.
The problem of taking station capacities into account is also relevant in timetabling. Here, one has to check for a given timetable whether the capacity in every station is sufficient. Instead of considering the number of platforms as the capacities of the stations, it is even more realistic to look at the *train routing problem*, *i.e.*, to find routes through the stations for all the trains using the detailed track topology. This feasibility problem has been extensively studied. In Kroon et al. (1997), a set of inbound and outbound routes is given for each train. If a train chooses one of these routes, all track sections of it are reserved at once but released section-wise. It is shown that deciding whether a feasible schedule exists is NP-complete already for three possible routes per train. In Caprara et al. (2011), the problem is modeled as an integer program using clique inequalities in a conflict graph. For a recent survey on railway track allocation problems, see Lusby et al. (2011).

3 Integer Programming Formulations

In this section we present an integer programming formulation for the delay management problem that takes the capacities within stations into account. As basis for this model we use the integer programming formulation that includes capacities of the tracks as it was introduced in Schachtebeck and Schöbel (2010). Note that other formulations of the DM problem can analogously be extended to take the stations' capacities into account. We now first describe the integer programming formulation without station capacities and then show how to incorporate the limited capacity of the station infrastructure.

3.1 Formulation without station capacities

For modeling delay management problems as integer programs, usually an *event-activity network* $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ is used as underlying directed graph. The set of nodes $\mathcal{E}$ corresponds to the arrival and departure events of all trains at all stations. In Figure 3, an event-activity network is depicted for two trains. The lower train is a long-distance train traveling from s_1 to s_4 via s_3. The upper train is a regional train that also travels from s_1 to s_3, but stops at station s_2 also. After s_3, it continues in the direction of s_5. The events are represented as rectangles in

the picture. The set $\mathcal{A}$ consists of the following activities. Between the arrival i and the departure j of a train in the same station, there is a *waiting activity* $a = (i, j) \in \mathcal{A}_{\text{wait}}$; between a departure i of a train in a station and its arrival j in the next station there is a *driving activity* $a = (i, j) \in \mathcal{A}_{\text{drive}}$. The set $\mathcal{A}$ furthermore contains *transfer activities* $\mathcal{A}_{\text{transfer}}$ linking an arrival of a train in a station to a departure of another train in the same station. In Figure 3, there is a transfer at s_3 from the regional to the long-distance train, which is depicted by a dashed arrow. Finally, *headway activities* are needed for any pair of trains competing for the same infrastructure. These headway activities represent pairs of precedence relations of which one must be selected. To illustrate this, let i and j be two departures that continue over a common track. We denote the corresponding arrivals at the next station by i' and j', respectively. If departure i takes place before departure j, then arrival i' should also be scheduled before arrival j'. This choice for the order of the trains is represented by a pair of headway activities $a_1 = (i, j), a_2 = (j, i) \in \mathcal{A}_{\text{head}}$. For each pair of headway activities, we define both a pair of constraints for the departures of the trains and a pair for the arrival of the trains. In our example, these two pairs of constraints corresponding to *one* pair of headway activities are shown with dotted arcs. Each activity $a \in \mathcal{A}$ requires a minimal duration that is denoted by L_a.

The most important decision in delay management is which connections need to be maintained. For each changing activity $a \in \mathcal{A}_{\text{transfer}}$ we thus introduce a binary decision variable z_a, which is defined as follows.

$$z_a = \begin{cases} 0 & \text{if connection } a \text{ is maintained,} \\ 1 & \text{otherwise.} \end{cases}$$

In order to take the capacity constraints on the tracks into account, one defines a binary decision variable g_{ij} for each headway activity $(i, j) \in \mathcal{A}_{\text{head}}$, given as

$$g_{ij} = \begin{cases} 0 & \text{if event } i \text{ takes place before event } j, \\ 1 & \text{otherwise.} \end{cases}$$

For each event $i \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$, we define $x_i \in \mathbb{N}$ as the rescheduled time when event i takes place. The set of variables $x = (x_i)$ defines the disposition timetable. If the wait-depart decisions z_a and the priority decisions g_{ij} are fixed, the values of x_i, $i \in \mathcal{E}$ can easily be calculated by the critical path method (see Schöbel (2006)).

Given the original timetable π_i, $i \in \mathcal{E}$ and a set of exogenous source delays d_i at events and d_a at activities (being zero if there is no delay), the integer programming formulation (DM) without station capacities reads as follows.

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

such that

$$x_i \geq \pi_i + d_i \qquad \forall i \in \mathcal{E}, \tag{2}$$

$$x_j - x_i \geq L_a + d_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}}, \tag{3}$$

$$Mz_a + x_j - x_i \geq L_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{transfer}}, \tag{4}$$

$$Mg_{ij} + x_j - x_i \geq L_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{head}}, \tag{5}$$

$$Mg_{ij} + x_{j'} - x_{i'} \geq L_a \qquad \forall a = (i,j) \in \mathcal{A}_{\text{head}}, \tag{6}$$

$$g_{ij} + g_{ji} = 1 \qquad \forall (i,j) \in \mathcal{A}_{\text{head}}, \tag{7}$$

$$x_i \in \mathbb{N} \qquad \forall i \in \mathcal{E}, \tag{8}$$

$$z_a \in \{0,1\} \qquad \forall a \in \mathcal{A}_{\text{change}}, \tag{9}$$

$$g_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}_{\text{head}}. \tag{10}$$

The objective function in this model counts the sum of delays of all events (weighted with the number of passengers c_i who arrive at their final destination at event i) and adds a penalty of T for every passenger who misses a connection. In a periodic timetable, T is often chosen as its cycle time. We weigh the transfer activity a with the number of passengers c_a who planned to use it. The objective is an approximation of the overall delay of all passengers and is commonly used in delay management. It gives the exact value if the never-meet property for headways holds (see Schachtebeck and Schöbel (2010)). A more realistic model taking into account the real paths that passengers would use in case of delays has been developed in Dollevoet et al. (2012b). It can also be used as basis for our extension, but it is technically more difficult and computationally harder to solve.
The interpretation of the constraints is as follows. (2) makes sure that trains do not depart earlier than planned and that source delays at events are taken into account. (3) propagates the delay along waiting and driving activities while (4) propagates the delay along *maintained* changing activities. For each pair of departure events competing for the same infrastructure, (7) makes sure that exactly one of the two precedence relations is respected. (5) propagates the delay along the corresponding headway activity between the departures of the trains. Similarly, (6) propagates the delay between the arrivals of the trains. Here i' and j' are the arrivals that follow the departures i and j, respectively.

3.2 Formulation with a dynamic platform assignment

To include the limited capacity within the stations, we now present a formulation for delay management which allows a dynamic assignment of trains to platform tracks. Preliminary computational results showed that this assignment-based formulation performs much better than a packing-based formulation modeling the same problem (see Dollevoet et al. (2011)). Our assignment-based integer programming formulation views a station as a set of platforms, and introduces headway constraints for trains that make use of the same platform track. As a consequence, this formulation determines an explicit allocation of the events to the available platforms.

In order to allocate the trains to the platforms, we first define for each station $s \in S$ the set P_s of platforms at s and the set $\mathcal{E}^s_{\text{arr}}$ of arrival events at s. Then, we introduce binary decision variables y_{ip} for each event $i \in \mathcal{E}^s_{\text{arr}}$ and $p \in P_s$, that are defined as

$$y_{ip} = \begin{cases} 1 & \text{if arrival } i \text{ and corresponding departure are assigned to platform track } p, \\ 0 & \text{otherwise.} \end{cases}$$

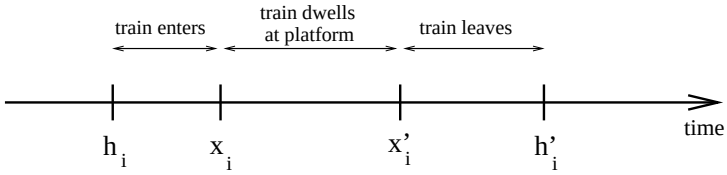

Figure 4: Illustration of the enter time, the leave time and the time during which the platform track is occupied.

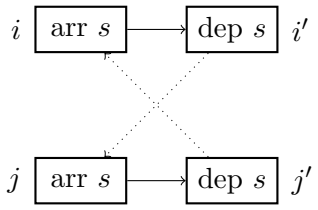

Figure 5: When two trains use the same platform track, a pair of headway activities is introduced from the departure of one train to the arrival of the other.

Of course, each arrival event must be assigned to exactly one platform track. This is enforced by the following constraint.

$$\sum_{p \in P_s} y_{ip} = 1, \quad \forall s \in S, i \in \mathcal{E}^s_{\text{arr}}. \tag{11}$$

In order to model the limited capacity of the stations, we determine the order in which the trains arrive at a certain platform track. Consider a pair of trains (t_1, t_2) that arrive at the same station corresponding to two events i and j. If the two trains are assigned to the same platform track, we must determine the order in which the events i and j take place. To this end, we introduce a pair of binary variables $\bar{g}_{ij}$ and $\bar{g}_{ji}$ that are defined as follows

$$\bar{g}_{ij} = \begin{cases} 0 & \text{if arrival } i \text{ takes place before arrival } j \text{ on the same platform track,} \\ 1 & \text{otherwise.} \end{cases}$$

If the trains are assigned to the same platform track, either t_1 must have departed before t_2 arrives, or t_2 must have departed before t_1 arrives. It should be noted that a train starts entering a station at a time h_i before it stops there at time x_i and passengers can board. The time h_i when the train starts to enter the station is called *enter time*. In the same way, the departure time $x_{i'}$ of a train is smaller than the *leave time* $h_{i'}$, which is the time when the last car of the train leaves the platform track and hence the time when the next train can start to enter. Thus $[h_i, h_{i'}]$ denotes the interval during which a platform track is occupied (see Figure 4).

We define $l_i = x_i - h_i$ for arrival events and $l_{i'} = h_{i'} - x_{i'}$ for departure events. By construction, l_i and $l_{i'}$ are non-negative. We define the headway time $L_{ij} = l_i + l_{j'}$ as the time that is minimally needed between the departure i and the arrival j.

In Figure 5 the event-activity network for the trains t_1 and t_2 is depicted. Let $a_i = (i, i')$ be the waiting activity of train t_1 and let $a_j = (j, j')$ be the waiting activity of train t_2. We define a pair of platform track activities $a_1 = (i', j), a_2 = (j', i) \in \mathcal{A}_{\text{plat}}$ and introduce the following set of constraints.

$$M\bar{g}_{ij} + x_j - x_{i'} \geq L_{ij} = l_{i'} + l_j, \tag{12}$$

$$M\bar{g}_{ji} + x_i - x_{j'} \geq L_{ji} = l_{j'} + l_i, \tag{13}$$

$$\bar{g}_{ij} + \bar{g}_{ji} \leq 3 - y_{ip} - y_{jp} \quad \forall p. \tag{14}$$

These constraints can be interpreted in the following way. Assume first that trains t_1 and t_2 are not assigned to the same platform track. Then $3 - y_{ip} - y_{jp} \geq 2$ for all p. Hence, both $\bar{g}_{ij}$ and $\bar{g}_{ji}$ can be set to 1 and (12) and (13) are satisfied. Otherwise, if trains t_1 and t_2 are assigned to the same platform track p, then $3 - y_{ip} - y_{jp} = 1$ for that p, forcing either $\bar{g}_{ij}$ or $\bar{g}_{ji}$ to zero. In that case, one of the headway constraints enforces a minimal headway time between the two trains.

The above constraints must be introduced for each pair of trains (t_1, t_2) that dwell at a common station $s \in S$. Recall that the set of platform track activities is denoted by $\mathcal{A}_{\text{plat}}$. This formulation thus introduces one binary variable $\bar{g}_{ij}$ for each $a = (i, j) \in \mathcal{A}_{\text{plat}}$ and one binary variable y_{ip} for each $i \in \mathcal{E}_{\text{arr}}$ and $p \in P_s$, where $s \in S$ is the station corresponding to arrival i. Furthermore, it adds one constraint for each arrival event $i \in \mathcal{E}_{\text{arr}}$ and $2 + |P_s|$ constraints for each pair of trains (t_1, t_2) that dwell at a common station $s \in S$. Note that this type of constraints are referred to as blocking constraints in the context of job-shop scheduling (see Mascis and Pacciarelli (2002)).

Adding the constraints (11)-(14), $y_{ip} \in \{0, 1\}$ for all $s \in S$, $i \in \mathcal{E}^s_{\text{arr}}, p \in P_s$ and $\bar{g}_{ij} \in \{0, 1\}$ for all $(i, j) \in \mathcal{A}_{\text{plat}}$ to the formulation (1)-(10) we obtain an integer programming formulation (DM-Cap) for the delay management problem with a *dynamic platform assignment*.

3.3 Formulation with a static platform assignment

Note that the planned timetable provides us with a platform assignment. To avoid platform track changes for the passengers and, at the same time, simplify our calculations, we could fix this platform assignment, *i.e.*, the variables y_{ip}. More generally, we call the delay management problem for which a platform assignment is determined in advance *delay management with a static platform assignment*. For delay management with a static platform assignment, the above integer programming formulation reduces to a problem of type (DM).

Lemma 1. *For fixed variables y_{ip} for all $i \in \mathcal{E}^s_{arr}, p \in P_s$ the formulation (DM-Cap) reduces to an instance of (DM),* i.e.*, can be solved as a delay management problem with headway constraints.*

Proof. If all y_{ip} variables are fixed we have two possibilities for (14): Either both y_{ip} variables are 1, then $\bar{g}_{ij} + \bar{g}_{ji} \leq 1$ and (12)-(13) reduce to a headway constraint of type (5)-(7), or at least one of the y_{ip} variables is 0, then (12)-(14) becomes redundant. □

According to this lemma, we can derive the following two bounds for delay management with a dynamic platform assignment, which can easily be calculated using an algorithm that solves problem (DM). First, it is clear that (DM) is a relaxation of (DM-Cap), hence its objective value z^{DM} is a lower bound. Second, if we fix the assignment y of trains

to stations in (DM-Cap) and solve the delay management problem with a static platform assignment, we obtain an upper bound $z^*(y)$ which can also be calculated by any algorithm for (DM) according to Lemma 1. Hence, we can compute an upper and a lower bound, *i.e.*, $z^{DM} \leq z^* \leq z^*(y)$.

4 An iterative approach

In the previous section we developed a model that simultaneously optimizes the platform assignment and the priority decisions in the stations. It is well known in timetabling that this problem is computationally challenging. As we consider a real-time setting, solutions to the delay management problem should be available within a short computation time. For large instances, optimizing the wait-depart decisions, the priority decisions and the platform assignments simultaneously might be intractable. For these instances, we propose an iterative approach: We first fix the assignment of trains to platforms as given in the original timetable. This results in a problem of type (DM) which can be solved according to Schachtebeck and Schöbel (2010). For the resulting solution we then try to improve the platform assignment within the stations and iterate until no further improvement is found. Using formulation (DM-Cap) we obtain:

1. Fix the station assignment y_{ip} in (DM-Cap) according to the planned timetable.
2. Solve the resulting problem (DM-Cap) with fixed y_{ip} and obtain a solution with disposition timetable x_i, wait depart decisions z_a and priority decisions g_{ij} and $\bar{g}_{ij}$
3. For every station, determine a more promising platform assignment y_{ip} and new priority decisions $\bar{g}_{ij}$ within the station such that $(x, z, y, g, \bar{g})$ is feasible.
4. Go to Step 2. Stop if no further improvement has been found.

If for large delay management instances decomposing the problem into two steps still results in long running times, we can use the approach of Schachtebeck (2010) to decompose Step 2 of the algorithm further into two smaller subproblems making first the priority decisions and then the wait-depart decisions.
In Step 3, a natural idea would be to adjust not only the platform assignment but also the timetable locally. Unfortunately, this can lead to infeasible solutions. Therefore, in Step 3 of the algorithm, we leave the timetable unchanged and adjust only the platform assignment in a way that allows the subsequent delay management step to shift events forward in time, if possible.

In the following we discuss Step 3, *i.e.*, how to find an assignment of trains to platforms at a given station s which is feasible for the given disposition timetable x and potentially yields a better disposition timetable in Step 2 of the next iteration. Recall from (12) and (13) that the headway times L_{ij} between two trains are the sum of a headway time $l_{i'}$ that is needed for the first train to leave the station after its departure event i' and a headway time l_j representing the time that the second train needs to completely enter the station before its arrival event j can take place, *i.e.*, $L_{ij} = l_{i'} + l_j$. Thus instead of scheduling the arrival and departure events x_i, we can instead schedule the enter time $h_i = x_i - l_i$ for arrival events i and the leave time $h_{i'} = x_{i'} + l_{i'}$ for departure events i' in a way that the intervals $(h_i, h_{i'})$ and $(h_j, h_{j'})$ do not overlap for two trains with arrival and departure events i, i' and j, j', respectively, that are

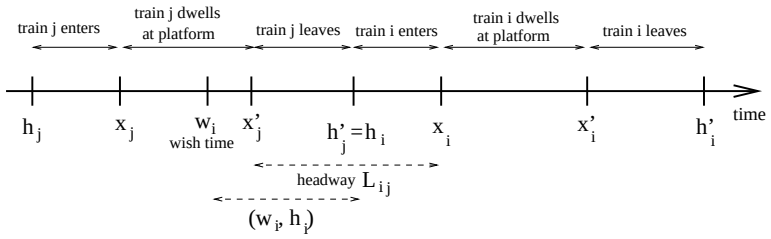

Figure 6: Illustration for two trains occupying the same platform track.

assigned to the same platform track. We process each station separately as follows. In a first step we identify for which arrivals $i \in \mathcal{E}$ in this station a new assignment might be beneficial. These are arrivals of delayed trains that directly follow another delayed train. For these train arrivals we determine their *wish (enter) times* w_i. In a second step we find a new assignment for all trains together with new enter times $q_i \geq w_i$ for these trains which should be as close to the wish times as possible. We now first show how the *wish times* are identified.
Let P_s be the set of platforms and $\mathcal{E}^s_{\text{arr}}$ be the set of arrival events in station s. Note that every such event corresponds to one train. For every arrival event i, let i' be the departure event following i (*i.e.*, $(i, i') \in \mathcal{A}_{\text{wait}}$ describes the waiting activity of the train in the station). From the timetable and the headway times we know that the train occupies the station during the time interval $(h_i, h_{i'})$. If a train is delayed, we distinguish two cases (see Figure 6).

- There is another train which is in the station during the interval $(h_j, h_{j'})$ with $h_{j'} = h_i$, and is on the same platform track p, *i.e.*, $y_{ip} = y_{jp} = 1$. In this case, a new assignment might help to reduce the delay of i. Assuming that $a = (k, i) \in \mathcal{A}_{\text{drive}}$ is the preceding driving activity of the train we define the *wish time* of i as

$$w_i := x_k + L_a + d_a - l_i.$$

- If no other train is on the same platform track directly before x_i, the delay of i is not due to the station assignment, and hence $w_i := h_i$.

Also if the train is not delayed we set $w_i := h_i$. The *platform assignment* problem (PA) can now be formulated as follows.

(PA) *Given a set of platforms $P_s = \{1, \ldots, P\}$ and for every arrival event $i \in \mathcal{E}^s_{arr}$ an interval $[h_i, h_{i'}]$ and a wish time $w_i \leq h_i$ as well as a weight c_i corresponding to the affected customers on the train, find numbers $q_i \in [w_i, h_i]$ for all $i \in \mathcal{E}^s_{arr}$ and a new platform assignment y_{ip} for all $i \in \mathcal{E}^s_{arr}$ and $p \in P_s$ such that*

$$q_j \in (q_i, h_{i'}) \Longrightarrow y_{ip} + y_{jp} \leq 1 \tag{15}$$

holds for all $i, j \in \mathcal{E}^s_{arr}$ and $p \in P_s$ and $\sum_{i \in \mathcal{E}^s_{arr}} c_i q_i$ is minimal.

Note that $q_j \in (q_i, h_{i'})$ or $q_i \in (q_j, h_{j'}) \iff (q_i, h_{i'}) \cap (q_j, h_{j'}) \neq \emptyset$, *i.e.*, the two trains belonging to i and j cannot be scheduled on the same platform track if and only if the arrival of one train is scheduled at a time when the other train is occupying the platform track.

This problem can be formulated as a mixed-integer program as it is but the formulation does not seem to be promising due to condition (15). Instead we show that (PA) is polynomially solvable by first identifying a finite dominating set $\mathcal{C}$ for the q_i variables. This set $\mathcal{C}$ contains a polynomial number of elements. We then notice that for every choice of the q_i variables, we can check feasibility by solving a coloring problem. Naively, in order to check all possible $q \in \mathcal{C}^{|\mathcal{E}^s_{\text{arr}}|}$, we would have to solve an exponential number of coloring problems. Instead, we use that the graph under consideration is an interval graph and code the solvability of the coloring problem in the constraints of an IP formulation. This problem can be solved easily, as the coefficient matrix is totally unimodular. We first identify a dominating set $\mathcal{C}$ with a polynomial number of elements. In order to reduce the problem size, we also show that for each q_i, a smaller dominating set $\mathcal{C}_i$ can be defined.

Lemma 2. *Let $\mathcal{C} := \bigcup_{i \in \mathcal{E}^s_{arr}} \{w_i, h_i, h_{i'}\}$ be the set of all given wish and planned arrival and departure times. Then there exists an optimal solution (q, y) to (PA) with $q_i \in \mathcal{C}_i := \mathcal{C} \cap [w_i, h_i]$ for all $i \in \mathcal{E}^s_{arr}$.*

Proof. Let (q, y) be a feasible solution to (PA). Clearly, $w_i \le q_i \le h_i$ for all i. Furthermore, with p the platform track for which $y_{ip} = 1$, $q_i \ge \max\{h_{j'} : y_{jp} = 1 \text{ and } h_{j'} \le q_i\}$. Now assume that $q_i \notin \mathcal{C}$ for some $i \in \mathcal{E}^s_{\text{arr}}$. Let p be the platform track with $y_{ip} = 1$. Define

$$\tilde{q}_i := \max\left\{w_i, \max\{h_{j'} : y_{jp} = 1 \text{ and } h_{j'} \le q_i\}\right\}. \tag{16}$$

Then $\tilde{q}_i \in [w_i, h_i]$ and for all j condition (15) is still satisfied. Hence, replacing q_i by $\tilde{q}_i$ is a feasible solution to (PA) with better objective value and with $\tilde{q}_i \in \mathcal{C}_i$. Doing this for all values q shows the result. □

Now assume that some values $q_i \in \mathcal{C}_i$, $i \in \mathcal{E}^s_{\text{arr}}$ are given. How can we check whether q is feasible? This means that we have to check whether there is a platform assignment y such that (15) is satisfied. To this end we transform our problem into a coloring problem in the graph $G(q) = (\mathcal{E}^s_{\text{arr}}, E)$. For every $i \in \mathcal{E}^s_{\text{arr}}$ there exists a node. We add an edge $\{i, j\}$ between two nodes if $(q_i, h_{i'}) \cap (q_j, h_{j'}) \neq \emptyset$, *i.e.*, if the two corresponding trains cannot be assigned to the same platform track. In order to find out whether there is a feasible platform assignment for q we thus have to find out whether $G(q)$ is P-colorable. Note that by construction this graph is an interval graph and thus perfect (see *e.g.* Schrijver (2003)). Thus $\chi(G(q)) = \omega(G(q))$ with $\chi(G(q))$ denoting the chromatic number of $G(q)$ and $\omega(G(q))$ the number of nodes in the biggest clique of $G(q)$. We hence have to check whether the number of nodes in the biggest clique in $G(q)$ is not greater than P.

Let us order the values in $\mathcal{C} = \{q^1, \ldots, q^{|\mathcal{C}|}\}$ in increasing order and let us define intervals $I_l := (q^l, q^{l+1})$ for $l = 1, \ldots |\mathcal{C}| - 1$. For a given q we define a matrix $A(q) = (a_{li})$ with $|\mathcal{C}| - 1$ rows and $|\mathcal{E}^s_{\text{arr}}|$ columns and entries

$$a_{li} = \begin{cases} 1 & \text{if } (q_i, h_{i'}) \cap I_l \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Then we can determine the chromatic number of $G(q)$ as follows.

Lemma 3.

$$\omega(G(q)) = \max_{l=1,\ldots,|\mathcal{C}|-1} \sum_{i \in \mathcal{E}^s_{arr}} a_{li}.$$

Proof. Due to Lemma 2 we can assume that all values of q_i are in $\mathcal{C}$, hence there is an edge between i and j in $G(q)$ if and only if there exists an interval I_l such that $a_{li} = a_{lj} = 1$. Now let $\mathcal{E}' \subseteq \mathcal{E}^s_{\text{arr}}$. As $G(q)$ is an interval graph, $\mathcal{E}'$ is a clique in $G(q)$ if and only if there exists *one* interval I_l such that $a_{li} = 1$ for all $i \in \mathcal{E}'$. □

Now we can finally rewrite (PA) as an integer program in which we look for a choice of q-values from the set $\mathcal{C}$ checking feasibility by Lemma 3. Denote by q_i^k the entries of the set $\mathcal{C}_i = \{q_i^1, q_i^2, \ldots, q_i^{|\mathcal{C}_i|}\}$. Then for every arrival event i and every candidate $q_i^k \in \mathcal{C}_i$ we define the variable

$$\eta_i^k = \begin{cases} 1 & \text{if candidate } q_i^k \in \mathcal{C}_i \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$$

These are the variables of our integer program. In order to directly see properties of the resulting constraint matrix, we order our variables such that all variables η_i^k having the same index i are grouped together. We need to extend the matrix defined in (17) to all possible choices of q. To this end, we define for every (i,k) a column with

$$\tilde{a}_{lik} = \begin{cases} 1 & \text{if } (q_i^k, h_{i'}) \cap I_l \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Doing this for all $i = 1, \ldots, |\mathcal{E}^s_{\text{arr}}|$ we obtain a matrix $\tilde{A} = (\tilde{a}_{lik})$ with $|\mathcal{C}| - 1$ rows and $\sum_{i \in \mathcal{E}^s_{\text{arr}}} |\mathcal{C}_i|$ columns. Note that $q_i^k = q_j^{k'}$ with $q_i^k \in \mathcal{C}_i$, $q_j^{k'} \in \mathcal{C}_j$ is possible but would lead to two (maybe different) columns in $\tilde{A}$.

(PA) can hence be rewritten as

$$\min \sum_{i \in \mathcal{E}^s_{\text{arr}}} c_i \sum_{k=1}^{|\mathcal{C}_i|} q_i^k \eta_i^k \tag{18}$$

such that

$$\sum_{k=1}^{|\mathcal{C}_i|} \eta_i^k = 1 \qquad \forall i \in \mathcal{E}^s_{\text{arr}}, \tag{19}$$

$$\sum_{i \in \mathcal{E}^s_{\text{arr}}} \sum_{k=1}^{|\mathcal{C}_i|} \tilde{a}_{lik} \eta_i^k \leq P \qquad l = 1, \ldots, |\mathcal{C}| - 1, \tag{20}$$

$$\eta_i^k \in \{0,1\} \qquad \forall i \in \mathcal{E}^s_{\text{arr}},\ \forall k \in \mathcal{C}_i. \tag{21}$$

Lemma 4. *The constraint matrix A' defined by the inequalities (19)-(20) is totally unimodular.*

Proof. To avoid notational confusion, for the matrix A' we use u as an index for the rows and v as an index for the columns. We show that A' is totally unimodular by showing that every subset U of rows of A' can be partitioned into two sets U_1, U_2 with $U_1 \cap U_2 = \emptyset$, $U_1 \cup U_2 = U$ and $\sum_{u \in U_1} a'_{uv} - \sum_{u \in U_2} a'_{uv} \in \{-1, 0, 1\}$ for all columns v (see for example Schrijver (2003)).

The columns of A' are associated with the variables of our integer program. For every $i = 1, \ldots, |\mathcal{E}^s_{\text{arr}}|$ we denote by $C(i)$ the indices v of the columns of A' associated with a variable η_i^k for some k.

The rows represent the constraints. The first rows $u = 1, \ldots, |\mathcal{E}^s_{\text{arr}}|$ contain the constraints that for every $i \in \mathcal{E}^s_{\text{arr}}$, exactly one variable η^k_i is set to 1. We thus have

$$a'_{uv} = \begin{cases} 1 & \text{if the column } v \text{ belongs to variable } \eta^k_u \text{ for a } k, \textit{i.e.}, \text{ if } v \in C(u), \\ 0 & \text{otherwise.} \end{cases}$$

for $u = 1, \ldots, |\mathcal{E}^s_{\text{arr}}|$, *i.e.*, for u corresponding to a constraint for an $i \in \mathcal{E}^s_{\text{arr}}$.

Starting from row $|\mathcal{E}^s_{\text{arr}}| + 1$, the matrix A' consists of the matrix $\tilde{A}$. We notice that $\tilde{A}$ has the column-wise consecutive ones property. Furthermore, we note that every column $v \in C(i)$ of A' has its last 1-entry in the row that represents the constraint for the interval with end point h_i.

Let U be an index set of rows of A' and $U^A = U \setminus \{1, \ldots, |\mathcal{E}^s_{\text{arr}}|\}$, that is the part of the chosen set of rows that is contained in $\tilde{A}$. We alternatingly assign the rows in U^A to two sets U^A_1 and U^A_2. Then for every $i \in \mathcal{E}^s_{\text{arr}}$ either

$$\left\{ \sum_{u \in U^A_1} a'_{uv} - \sum_{u \in U^A_2} a'_{uv} : v \in C(i) \right\} \subseteq \{-1, 0\} \tag{22}$$

$$\text{or } \left\{ \sum_{u \in U^A_1} a'_{uv} - \sum_{u \in U^A_2} a'_{uv} : v \in C(i) \right\} \subseteq \{1, 0\} \tag{23}$$

because of the consecutive ones property and because for a given i, the last entry of column v is in the same row for all $v \in C(i)$.

We set $U_1 := U^A_1$ and $U_2 := U^A_2$ and add the indices of the first $|\mathcal{E}^s_{\text{arr}}|$ rows in the following way to these sets: If for row u (22) holds, we assign the u-th row to U_1, if (23) holds we assign it to U_2. We obtain

$$\left\{ \sum_{u \in U_1} a'_{uv} - \sum_{u \in U_2} a'_{uv} : v \in C(i) \right\} \subseteq \{1, 0\} \text{ for all } i \in \{1, \ldots, |\mathcal{E}^s_{\text{arr}}|\} \text{ with (22)}$$

$$\left\{ \sum_{u \in U_1} a'_{uv} - \sum_{u \in U_2} a'_{uv} : v \in C(i) \right\} \subseteq \{-1, 0\} \text{ for all } i \in \{1, \ldots, |\mathcal{E}^s_{\text{arr}}|\} \text{ with (23).}$$

This proves total unimodularity. □

Corollary 1. *(PA) can be solved by linear programming.*

We conclude that the problem in Step 3 of the iterative algorithm can be solved by linear programming. This completes the description of the iterative algorithm.

5 Computational results

We have performed a computational study to test whether it is important to consider the capacity within stations explicitly and to compare the different approaches presented in this paper. We first describe the cases that were used in this study. Then we show that a dynamic platform assignment significantly improves a static one. Finally, we evaluate the performance of the iterative heuristic.

5.1 Cases

In our numerical experiments we consider the railway system in the Randstad, which is the mid-Western part of the Netherlands. Figure 1 gives a schematic representation of the railway network in this region. The dots in this figure indicate a station where long-distance trains stop. The stations where only regional trains stop are not depicted. A line indicates that there is a direct link between two stations. For each link, there are two or four long-distance trains and two regional trains per hour. It can be seen in the picture that the railway network contains direct links between many of the stations. As a consequence, the infrastructure is heavily utilized, especially in the stations.

We have generated four cases, that vary in the size of the network and the type of trains that are included. The first case considers only the stations in the network that are indicated by a black dot in Figure 1 and includes only the long-distance trains. Case B considers the same network, but includes both long-distance and regional trains. The third and fourth case include all stations that are indicated by a black or a white dot. Again, Case C considers only the long-distance trains, while Case D includes the regional trains, too. These cases resemble those that are used in Dollevoet et al. (2012b).

We obtained the timetable and detailed information on the passenger demand from Netherlands Railways. The passenger figures are not the real numbers, but have been scaled for secrecy. For each pair of stations in the network, we were given the average number of passengers who want to travel between these stations on a regular day. From these origin-destination figures we obtained the average number of passengers w_i who arrive at their destination station with arrival event $i \in \mathcal{E}_{\text{arr}}$ and the number of passengers w_a who use transfer $a \in \mathcal{A}_{\text{transfer}}$.

In order to evaluate the performance of our delay management models, we have simulated for each case 100 delay scenarios. These scenarios were constructed as follows. Each driving and dwell activity has a probability of 10% to be delayed. If the activity is delayed, the size of the delay is a uniformly distributed random variable between 1 and 10 minutes. Note that delays on activities are additive: If two consecutive driving activities are delayed, the delay of the train is at least the sum of the two delays. We did not include delays at events.

Table 1 gives an impression of the sizes of the instances. For each of the four cases we report the number of stations and trains in the railway network. Besides, we report the number of events $|\mathcal{E}|$ and headway activities $|\mathcal{A}_{\text{head}}|$ in the resulting event-activity network. The column $|\mathcal{A}_{\text{plat}}|$ gives the number of platform track activities. Recall that there is a *pair* of platform track activities for each pair of trains (t_1, t_2) that dwell at a common station. $|\mathcal{A}_{\text{plat}}|$ is therefore *twice* the number of times that Constraints (12)-(14) are added to formulation (1)-(10). Cases B and D consider all trains in a large part of the network. Cases of these sizes arise in practical applications. Comparing Cases A and B, one sees that the number of trains is increased roughly by 50%. The number of nodes in the event-activity network is about 4 times as large. The reason is that the regional trains stop at far more stations than the long-distance trains. As priority decisions are only necessary at the larger stations, where overtaking can take place, the number of headway activities is related to the number of trains.

For each instance we have two different models to obtain a solution to the delay management problem with station capacities. Our first model fixes the platform assignment as planned. According to Lemma 1, this leads to a delay management problem with headway activities only. According to Section 3.3, we refer to this model as the static model. In the second,

			Size of the program			Static model		Dynamic model	
Case	Stations	Trains	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}_{\text{head}}\rvert$	$\lvert\mathcal{A}_{\text{plat}}\rvert$	Bin.	Con.	Bin.	Con.
A	10	82	344	1508	6962	1998	5470	9597	33924
B	34	193	1374	3432	28830	8045	21568	37972	175614
C	16	119	576	2296	11688	3345	8895	15913	55301
D	82	275	2804	5848	35138	12877	36510	48841	231101

Table 1: Some characteristics of the cases and the resulting integer programs. Bin. and Con. give the number of binary variables and constraints in the integer program, respectively.

	Objective value				Optimality gap			
Case	A	B	C	D	A	B	C	D
Static model	192291	1007849	595904	3635493	0%	0%	0%	0.3%
Dynamic model	192046	982983	585987	3572731	0%	0.1%	0%	1.3%

Table 2: The objective value and the optimality gap for the static and dynamic delay management model

dynamic model, we allow the platform tracks to be rescheduled dynamically as introduced in Section 3.2. In Table 1, we also list the number of binary variables and constraints in the resulting integer programs. One sees immediately that the number of binary variables and constraints is much larger for the dynamic model. As a consequence, the dynamic model is expected to be computationally much harder to solve.

5.2 Static and dynamic platform assignments

We have used Cplex 12.2 on an Intel Core i5-2410M with 4 GB of RAM to solve the integer programs from Section 3. The maximal computation time was set to 20 minutes for each individual delay scenario. Such times are too long for practical purposes, but allow us to find solutions that are close to optimal. The objective value for a case is computed as the average total delay over all scenarios. For each case, we compare the solutions that are obtained with a static and with a dynamic platform assignment.

In Table 2, the objective values are presented for both solution approaches. In Table 3, the relative improvement of the dynamic model is given, as well as the number of platform track changes in the solutions of the dynamic model. The results show clearly that rescheduling the platform assignment dynamically reduces the delay for the passengers. For Case A, the reduction is negligible, the average delay is reduced only by 0.1%. For Case B, the delay is

Case	A	B	C	D
Improvement	0.1%	2.5%	1.7%	1.7%
Platform track changes	167	342	269	380

Table 3: The improvement of the dynamic model with respect to the static model and the number of platform track changes in the solution of the dynamic model

Case	A	B	C	D
Static model	1.0 (1.6)	62.9 (1162)	2.0 (3.5)	765.6 (1204)
Dynamic model	3.1 (5.4)	608.7 (1353)	13.9 (112.7)	1408 (1478)

Table 4: The average running times and between brackets the maximal running times in seconds

reduced by 2.5%. This is a significant improvement over the static model. In the Cases C and D, the reduction is 1.7%. The optimality gap in the first three cases can be neglected. In Case D, the optimality gap for the dynamic model is 1.3%. The improvement of the dynamic model over the static model could therefore be larger than 1.7 %.
Besides the delay, platform track changes are also inconvenient for the passengers. The static model does not allow to reschedule the platform assignment, so in the static solutions there are no platform track changes. On the contrary, in the dynamic model a complete new platform assignment is determined. As we do not penalize changes in the platform assignment, the dynamic model introduced hundreds of platform track changes for every case.
As solutions to the delay management problem should be readily available, solution methods should be able to solve the delay management problem within a short computation time. In Table 4, the average and maximal running times are given for each case. Recall that we have set the maximal running time to 20 minutes for each delay scenario. For Cases A and C, which include only the long-distance trains, both models can be solved very fast. Both the average and maximal running time are less than two minutes. Such running times are acceptable in practice.
When the regional trains are included, the running times increase significantly. In order to speed up the solution process for Cases B and D, we first computed a solution to the static delay management model in which the order of trains in the stations and on the tracks is fixed. This results in an integer program that is much easier and can be solved within seconds. The solution to this model is also feasible for the dynamic model and can be used to decrease the solution times. In a similar fashion, a solution from the static model can be used when solving the dynamic model. When solving the dynamic model, we first ran the algorithm for the static model for 5 minutes. This explains why the maximal running times for the dynamic model are larger than 20 minutes.
For Case B, the static model can be solved within one minute on average. The dynamic model needs 10 minutes on average. For both models, some delay scenarios need much more computation time. For Case D, the static model can be solved within 12 minutes on average, while the dynamic model takes the full computation time of 25 minutes. In a real-time setting, such computation times are too long.

5.3 Performance of the iterative heuristic

In the previous section we have seen how a dynamic platform assignment can reduce the delay for the passengers. However, for larger cases, solving the dynamic model takes too much time. In these situations, the iterative heuristic from Section 4 can be applied to improve on the static solutions while still keeping the computation times within limits.

In Table 5, the results for the iterative heuristic are given. For all cases, the iterative heuristic finds solutions that are at most 1.0% worse than the dynamic model. For the cases

Case	A	B	C	D
Absolute objective	192088	992757	587307	3594245
Relative objective	100.0 %	101.0 %	100.2 %	100.6 %
Number of platform track changes	0.32	17.5	3.2	21.0
Number of iterations	2.06	4.14	2.72	4.81
Total solution time	2.2	174.1	4.76	1198

Table 5: The absolute objective value, the relative objective values with respect to the dynamic model, the number of platform track changes in the solutions and some characteristics of the iterative solution process

that consider only the long-distance trains, the iterative solutions are very close to the optimal ones. It is interesting to see that the iterative algorithm changes the platform assignment less often than the dynamic model. The iterative algorithm only schedules a train at a different platform track if it looks promising to do so. The slight decrease in quality is thus compensated by a platform assignment that looks much more like the original one.
Considering the computation times, the iterative approach solves the problem much faster than the dynamic model. For Cases A and C, the iterative algorithm can be executed within seconds. For Case B, the total running time of the iterative heuristic is on average less than 3 minutes. This is a reduction of 70% with respect to solving the dynamic model. A solution time of 3 minutes is acceptable in practice. For Case D, solving the static model to optimality already takes a longer time. We therefore allowed only 5 minutes of computation time per iteration. As a consequence, the first solution in the iterative procedure is slightly worse than the solution of the static model. The average total running time for Case D equals 20 minutes and is thus smaller than that of the dynamic model. It is, however, still too long for practical applications. We conclude that for cases with only long distance trains, both the performance and the running time of the dynamic model and the iterative algorithm are comparable. For Case B, solutions with slightly worse quality are obtained within much less computation time. For this case, the iterative algorithm finds good solutions within computation times that are acceptable in practice. Finally, for Case D, solutions are found that have a good quality, but the iterative algorithm needs too much time to find these solutions.

In Figure 7, we have plotted the progress of the iterative method. On the horizontal axis are the iterations, while the objective value is shown on the vertical axis. In order to show the progress for all cases in one figure, we have normalized the objective value. The objective value of the first iteration is equal to that of the static model. This value is indicated by the upper line, labeled with 1. A lower bound on the objective value from the iterative approach is given by the solutions of the dynamic model. This value is depicted by the lower line, labeled with 0.
For Cases A and C, the iterative algorithm improves the solution from the static model only in the first iteration. The solution that is then obtained is very close to the optimal solution. For Cases B and D, the biggest improvement is found in the first iteration. The improvement in the second iteration is much smaller. From then on, the solution does not change much. This suggests that one could also run only one iteration of the iterative algorithm, in order to improve over the static solution within a short computation time. For Case D, it takes on average 10 minutes to perform one iteration. In practical applications, computation times of 10 minutes are acceptable.

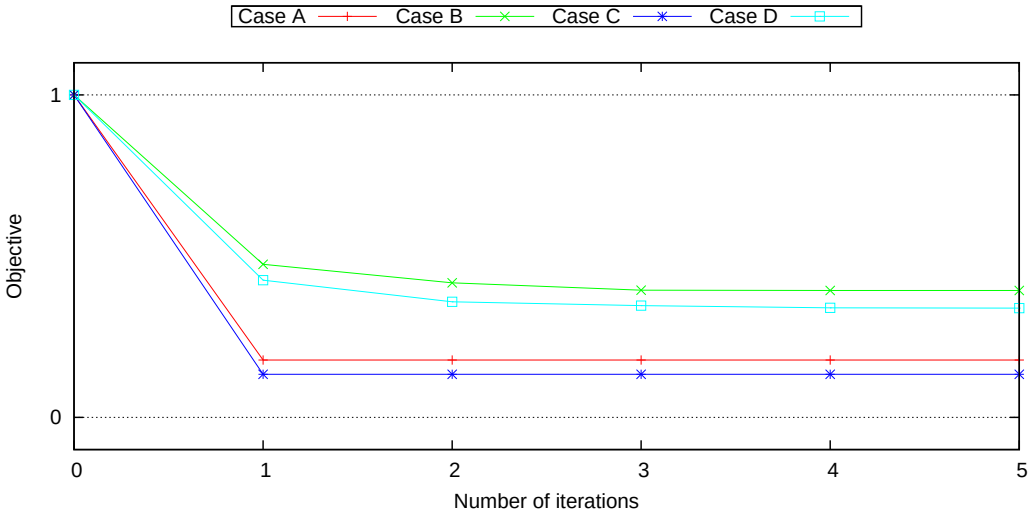

Figure 7: The progress of the iterative solution

5.4 Quality of the wait-depart decisions

The dynamic model optimizes the wait-depart decisions, the priority decisions and the platform assignment simultaneously. The original aim of delay management is, however, to decide on the wait-depart decisions only. In this section we will compare the quality of the wait-depart decisions from the dynamic model to those that are obtained with other delay management models from literature. If an optimization algorithm is available to determine the priority decisions and the platform assignment, the results in this section prescribe which delay management model should complement the train scheduling algorithm and optimize the wait-depart decisions.

In order to compare the quality of the different delay management models, we first decide on the wait-depart decisions with the various models. Then we fix the variables for the wait-depart decisions in the dynamic model in order to obtain the priority decisions and the platform assignment. We emphasize that we use the dynamic model only to compute the priority decisions and the platform assignment; the wait-depart decisions have been fixed by the model that we are interested in. We thus employ the dynamic model as a very basic train scheduling algorithm.

The first model that we consider implements a no-wait policy. With a no-wait policy, trains never wait for delayed trains. We use this policy as a benchmark for the other policies. The second model is a delay management model without any capacity considerations. This model is given by (1)-(4) and (8)-(9). The third model is a delay management model with priority decisions for tracks only. This model is presented in Section 3.1. Finally, the fourth model is the dynamic model that is introduced in Section 3.2.

In Table 6, the objective values are given for the four models. We have normalized the objective value and set the objective value of the dynamic model to 100. We see in the table that a no-wait policy performs very badly. The total delay is up to 40% higher than in the dynamic model. A delay management model without capacity considerations performs better. For Cases A and C, the delay is increased with 2%, while an increase of about 9% is observed for Cases B and D. Finally, the model that incorporates capacity constraints for tracks only performs very well. The maximal increase in delay is 0.3%. For the first three cases, the

Case	A	B	C	D
No-wait policy	120.1	140.4	124.5	134.4
DM without capacity constraints	102.6	108.7	101.7	108.7
DM with tracks priorities only	100.0	100.0	100.0	100.3
Dynamic model	100.0	100.0	100.0	100.0

Table 6: Comparing the dynamic model to other delay management models

increase is even negligible.
We conclude that the model that includes headway constraints on the tracks only finds wait-depart decisions that are very close to optimal. Furthermore, the model without station capacities is much simpler and solving it requires less computation time. When deciding on the wait-depart decisions, one thus need not consider the capacity within the stations explicitly.

6 Analyzing the trade off between passenger delays and platform track changes

In Section 5.2 we have seen that allowing a re-assignment of trains to platforms yields a significant improvement with respect to the passengers' delay, compared to the model where the platform assignment is fixed. On the one hand, this certainly increases the passengers' comfort. However, on the other hand, the improvement with respect to the delay is accompanied by many platform track changes with respect to the announced timetable. These platform track changes are certainly not convenient for the passengers. In that respect, the solutions from the iterative algorithm, with a worse objective value but less platform track changes, might be preferable in practice. Delay management with station capacities could hence be considered a bi-objective problem with the two objectives of minimizing

1. the passengers' delay, and
2. the number of changes in the platform assignment.

In Section 6.1 we discuss how this extension of our model can be integrated in the proposed solution approaches. Computational results are presented in Section 6.2.

6.1 Theoretical Modification of the models

We first consider the exact model for delay management with station capacities which is provided by the assignment-based integer programming formulation. We describe how a restriction on the number of platform track changes can be easily included.

Since in this formulation the information about the platforms where an event $i \in \mathcal{E}$ takes place is already coded in the variables y_{ip}, a restriction on the number of platform track changes can be modeled easily by adding *one* additional constraint. To this end, let C denote the maximal number of platform track changes and let $P(i)$ denote the set of *good platforms for event* i. If we want to count all platform track changes, $P(i)$ consist only of the platform track p_i where event i was scheduled initially. However, $P(i)$ could as well contain more platforms that are easily reached from p_i, *e.g.*, the track on the opposite side of the platform.

We add the following constraint to the integer program described in Section 3.2.

$$\sum_{i \in \mathcal{E}_{\text{dep}}} \sum_{p \notin P(i)} d_{ip} y_{ip} \leq C. \tag{24}$$

Here, d_{ip} is a weight that represents the importance of scheduling the departure event i on a good platform track. For example, d_{ip} could represent the number of passengers that have to move to another platform if the platform track is changed. Since y_{ip} takes the value 1 if event i is scheduled on platform track p and 0 otherwise, this constraint determines the weighted number of departure events which do not take place on a good platform track and restricts this number to C. This method is an example of the ϵ-constraint approach. If $d_{ip} = 1$ for all $i \in \mathcal{E}_{\text{dep}}$ and $p \notin P(i)$ and 0 otherwise, the summation in the left hand side just counts the number of departure events that are not scheduled at a good platform. This allows us to find all non-dominated solutions, as the number of platform track changes is discrete.

It is also possible to take into account the number of platform track changes in the iterative approach. When searching for a new platform assignment in the third step of the iterative algorithm, instead of minimizing the potential departure times only, we could additionally consider the number of platform track changes. However, since in the linear programming formulation (18)-(21) for the third step of the iterative approach we do not explicitly define an assignment to the platforms, for every candidate q_i^k we replace the variable η_i^k by a set of $|P_s|$ variables

$$(\eta_i^k)_p = \begin{cases} 1 & \text{if candidate } q_i^k \in \mathcal{C}_i \text{ is chosen and event } i \text{ takes place at platform track } p, \\ 0 & \text{otherwise.} \end{cases}$$

Then constraints (19-21) can be rewritten as

$$\sum_{p \in P_s} \sum_{k=1}^{|\mathcal{C}_i|} (\eta_i^k)_p = 1 \qquad \forall i \in \mathcal{E}^s_{\text{arr}}, \tag{25}$$

$$\sum_{i \in \mathcal{E}^s_{\text{arr}}} \sum_{k=1}^{|\mathcal{C}_i|} a^k_{il} (\eta_i^k)_p \leq 1 \qquad \forall l \in \{1, \ldots, |\mathcal{C}| - 1\},\ \forall p \in P_s \tag{26}$$

$$(\eta_i^k)_p \in \{0, 1\} \qquad \forall i \in \mathcal{E}^s_{\text{arr}},\ \forall k \in \mathcal{C}_i,\ \forall p \in P_s. \tag{27}$$

Analogously to the proof of Lemma 4 we have the following lemma.

Lemma 5. *The constraint matrix A' defined by the inequalities (25)-(26) is totally unimodular.*

In order to incorporate our second objective of minimizing the number of platform track changes, we could again add a constraint that restricts the number of platform track changes at each station. However, this requires a distribution of the C allowed platform track changes to the stations. Furthermore, the additional constraint in general destroys the property of total unimodularity and thus makes the problem much harder to solve.

Hence we include the minimization of the number of platform track changes in the objective function (28), considering a weighted sum of both objectives:

$$\min \sum_{i \in \mathcal{E}^s_{\text{arr}}} c_i \sum_{p \in P_s} \sum_{k=1}^{|\mathcal{C}_i|} q_i^k (\eta_i^k)_p + \lambda \sum_{i \in \mathcal{E}_{\text{dep}}} \sum_{k=1}^{|\mathcal{C}_i|} \sum_{p \notin P(i)} (d_i^k)_p (\eta_i^k)_p \tag{28}$$

where $(d_i^k)_p$ is set to 1 if we just want to penalize platform track changes but could also be modified to represent passenger weights or to penalize only the platform track changes of non-delayed trains.

The parameter λ can be used to control the influence of the different objective functions on the new platform assignment.

6.2 Computational Results

We have used the cases from Section 5.1 to compare the solution approaches that balance the delay on the one hand and the number of platform track changes on the other. For Cases A and C, we considered all 100 delay scenarios. For Cases B and D, we restricted ourselves to 30 delay scenarios to reduce the amount of computation time needed. We first present the results for the exact solution approach and then consider the iterative procedure.

In Figure 8, we have plotted the average delay for all passengers as a function of the maximal

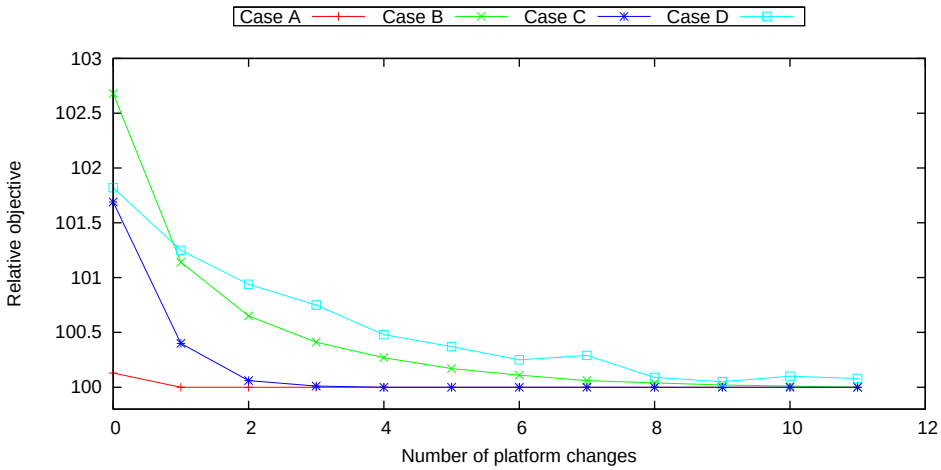

Figure 8: The average objective value as a function of the number of platform track changes that are allowed in the dynamic model.

number of platform track changes. For Cases A and C, the optimal solution can be obtained with at most 3 platform track changes for all delay scenarios. By allowing only 1 platform track change, the solutions for Case A are optimal. For Case C, the optimal solution with at most one platform track change are within 0.4% of optimality on average. For Case B, 9 platform track changes are needed to obtain the optimal solution. Again, more than half of the delay reduction can be obtained with only one platform track change. For Case D, the results look somewhat different. We again find the optimal solution with only 9 platform track changes. However, for Case D, the delay reduction with only one platform track change is relatively small. Furthermore, we observe that the solution with at most 10 platform track changes has a worse objective value than the solution with at most 9 platform track changes. Recall from Section 5.2 that the dynamic model cannot be solved to optimality for Case D. When we limit the number of platform track changes, the average optimality gap equals 0.5%. This explains the small increase in objective value when increasing the number of platform track changes from 9 to 10. We think the aberrant progress for Case D is also caused by our inability to solve the integer program to optimality.

In general, in order to find the optimal solution, much less platform track changes are needed

than were found by the dynamic model. Furthermore, the graph shows that a large part of the delay reduction can be obtained with only a small number of platform track changes.

In Figure 9, we have plotted the results for the iterative algorithm for Cases A and C. We have

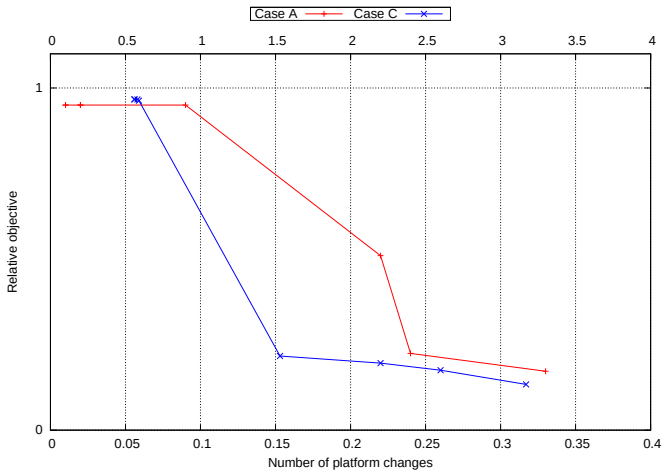

Figure 9: The average objective value and number of platform track changes for various values for the parameter λ. The lower x-axis corresponds to Case A; the upper x-axis to Case C

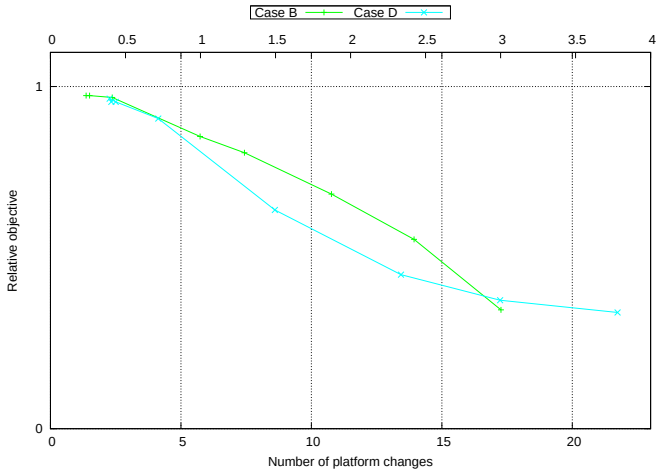

Figure 10: The average objective value and number of platform track changes for various values for the parameter λ for Cases B and D

run the algorithm for each value of the parameter $\lambda \in \{0.5, 1.5, 2.5, 4.5, 9.5, 19.5, 39.5, 99.5\}$. For each value of λ, we ran the iterative algorithm and computed both the total passenger delay and the number of platform track changes for each delay scenario. The total delay is normalized in the same way as in Figure 7. The averages of the total delay and platform track changes are plotted in the figure. Similarly, in Figure 10, the results are depicted for Cases B and D.

We see in these figures that the number of platform track changes can be reduced by incorporating them in our iterative algorithm. For low values of the parameter λ, we obtain the same objective value as with the original iterative algorithm, but find less platform track changes.

When the value of λ is increased, the number of platform track changes is reduced. However, this comes at the cost of more passenger delay.
For Case A, we find three interesting classes of solutions. The first class, found with $\lambda \geq 4.5$, resembles the static solution. Solutions in the second class, obtained for $\lambda \leq 1.5$, have an objective value that is comparable to that of the dynamic model, but introduce only few platform track changes. For $\lambda = 2.5$, a solution is found that balances the objective and the number of platform track changes. Note that for all values of λ, the average number of platform track changes is smaller than 1. This indicates that for most delay scenarios, there are no platform track changes at all. For Case C, only two classes of solutions are distinguished. For $\lambda \geq 9.5$, solutions are found with the same objective as the static approach. For $\lambda \leq 4.5$, we find an objective value that is close to the optimal objective, but the number of platform track changes is reduced significantly. For Cases B and D, the progress is more gradual. For all intermediate values of λ, a solution is found with a unique balance between the objective and the number of platform track changes.
Comparing both solution approaches for the bi-objective problem, we see that the exact algorithm finds solutions with less platform track changes. In terms of quality, the exact approach is thus preferable.
For Cases A and C, both the exact method and the iterative approach solve the instances within one minute. For Case B, the running time for the exact algorithm is in the order of 10 minutes, while the iterative approach can solve the model within 3 minutes. For practical applications, when computational time is scarce, it is thus better to apply the iterative algorithm. Finally, for Case D, both methods require 20 minutes of computation time. Such running times are too long for practical applications.

7 Conclusion and Further Research

In this paper, we introduced a delay management model that incorporates the limited capacity of railway stations. Two models are presented. The first model fixes the assignment of trains to platforms and reduces to a delay management model with headway constraints. In the second model it is allowed to reschedule the platform assignment. In a computational study, we show that the delay for the passengers can be reduced when the platform assignment is rescheduled dynamically.
As solutions to the delay management problem should be available within a very short computation time, we also proposed an iterative solution method for the delay management problem with station capacities. This heuristic iterates between solving a delay management problem with a given platform assignment and optimizing the platform assignment given the timetable and wait-depart decisions. We show that for each station separately, an optimal platform assignment can be found in polynomial time. Computational tests show that the iterative heuristic can be applied to improve on a solution that is obtained by the static delay management model, especially for cases that include regional trains.
A drawback of the dynamic model is that it reschedules a lot of trains to other platforms in order to reduce the total delay. Although delays are a source of frustration for the passengers, many platform track changes are frustrating, too. Furthermore, these track changes put pressure on the dispatching organization of the railway operator. In our view, delay management with station capacities should therefore be viewed as a bi-objective optimization problem. We show that much of the delay reduction can be obtained by allowing only a few platform

track changes. To resolve the remaining delays, many platform track changes are required. We therefore propose to limit the number of platform track changes that are allowed, in order to balance the delay for the passengers on the one hand and the number of platform track changes on the other.

We distinguish two directions for future research. The first direction searches for faster solution methods. Although our model can solve real-world instances within computation times that are allowed in practice, solution methods for even larger instances might be required. Furthermore, we approximate the delay for passengers who miss a connection by the cycle time T. In reality, these passengers will probably select an alternative route. To cope with this problem, our model can be used in an iterative solution approach as proposed by Dollevoet and Huisman (2011), but then it should be solved several times. In such a setting, faster solution methods are necessary. We think that further attempts to solve delay management with station capacities heuristically could be developed that make use of relaxation-based solution approaches.

The second direction applies our methods to the timetabling problem. Station capacities are also an important issue in timetabling. Therefore it would be interesting to apply our exact and heuristic solution methods to the timetabling problem with station capacities and to compare them to existing solution approaches.

References

A. Berger, R. Hoffmann, U. Lorenz, and S. Stiller. Online railway delay management: Hardness, simulation and computation. *Simulation*, 87(7):616–629, 2011.

G. Caimi, F. Chudak, M. Fuchsberger, M. Laumanns, and R. Zenklusen. A New Resource-Constrained Multicommodity Flow Model for Conflict-Free Train Routing and Scheduling. *Transportation Science*, 45(2):212–227, 2011.

A. Caprara, L. Galli, and P. Toth. Solution of the train platforming problem. *Transportation Science*, 45(2):246–257, 2011.

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

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1):175–192, 2010.

F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C: Emerging Technologies*, 20(1):79–94, 2012.

A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.

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

T. Dollevoet and D. Huisman. Fast heuristics for delay management with passengers rerouting. Technical report, Econometric Institute Report EI2011-35, Erasmus University Rotterdam, 2011.

T. Dollevoet, M. Schmidt, and A. Schöbel. Delay Management including Capacities of Stations. In A. Caprara and S. Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 88–99, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-33-0. doi: http://dx.doi.org/10.4230/OASIcs.ATMOS.2011.88. URL `http://drops.dagstuhl.de/opus/volltexte/2011/3269`.

T. Dollevoet, F. Corman, A. D'Ariano, and D. Huisman. An iterative optimization framework for delay management and train scheduling. Technical report, Econometric Institute Report EI2012-10, Erasmus University Rotterdam, 2012a.

T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1):74–89, 2012b.

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.

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*, volume 3787 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2005.

M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. Online delay management on a single train line. In F. Geraets, L. Kroon, A. Schoebel, D. Wagner, and C. Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2007.

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.

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

L. G. Kroon, H. E. Romeijn, and P. Zwaneveld. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3):485–498, 1997.

S. Krumke, C. Thielen, and C. Zeck. Extensions to online delay management on a single train line: new bounds for delay minimization and profit maximization. *Mathematical Methods of Operations Research*, 74(1):53–75, 2011.

R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883, 2011.

A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.

M. Schachtebeck. *Delay Management in Public Transportation: Capacities, Robustness, and Integration*. PhD thesis, Universität Göttingen, 2010.

M. Schachtebeck and A. Schöbel. To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, 44(3):307–321, 2010.

A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1):1–10, 2001.

A. Schöbel. *Optimization in public transportation. Stop location, delay management and tariff planning from a customer-oriented point of view*. Optimization and Its Applications. Springer, New York, 2006.

A. 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 *Lecture Notes in Computer Science*, pages 145–170. Springer, 2007.

A. Schöbel. Capacity constraints in delay management. *Public Transport*, 1(2):135–154, 2009.

A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg New York, 2003.