Delay Management including Capacities of Stations

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

1 **Econometric Institute and ECOPT, Erasmus University Rotterdam**
P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands
dollevoet@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

Abstract

The question of delay management (DM) is whether trains should wait for delayed feeder trains or should depart on time. Solutions to this problem strongly depend on the capacity constraints of the tracks making sure that no two trains can use the same piece of track at the same time. While these capacity constraints have been included in integer programming formulations for DM, the capacity constraints of the stations (only offering a limited number of platforms) have been neglected so far. This can lead to highly infeasible solutions. In order to overcome this problem we suggest two new formulations for DM both including the stations' capacities. We present numerical results showing that the assignment-based formulation is clearly superior to the packing formulation. We furthermore propose an iterative algorithm in which we improve the platform assignment with respect to the current delays of the trains at each station in each step. We will show that this subproblem asks for coloring the nodes of a graph with a given number of colors while minimizing the weight of the conflicts. We show that the graph to be colored is an interval graph and that the problem can be solved in polynomial time by presenting a totally unimodular IP formulation.

1998 ACM Subject Classification G.2.2 Network Problems

Keywords and phrases Delay Management, Station Capacities

Digital Object Identifier 10.4230/OASIcs.ATMOS.2011.88

1 Introduction and motivation

Passenger railway transport plays an important role in the European mobility. Especially during peak hours and for distances between 20 and 800 kilometers, passengers often choose to travel by train. In highly connected train systems passengers often have to change trains since it is impossible to give a direct connection between all origin-destination pairs. In order to minimize the inconvenience of changing from train A to train B, the timetable is often constructed in such a way that train B departs shortly after train A arrives. However, if train A has a delay during the operations, the question is whether train B should wait for train A or depart on time. Such decisions are called *delay management*. Delay management (DM) deals with (small) source delays of a railway system as they occur in the daily operations. In case of such delays, the scheduled timetable is not feasible any more and has to be updated to a *disposition timetable*. Note that since delays are often transferred if a connecting train waits for a delayed feeder train it is not clear in advance if it is an overall improvement for

11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems.
Editors: Alberto Caprara & Spyros Kontogiannis; pp. 88–99

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the system to maintain such connections. In order to ensure safe operations and to take the limited capacity of the track system into account, also *priority decisions* are necessary. They determine the order in which trains are allowed to pass a specific piece of track.

There exist various models and solution approaches for DM. 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 this problem has been given in [15] and has been further developed in [6] and [17]. The complexity of the problem has been investigated in [8] where it turns out that the problem is NP-hard even in very special cases. Recently, re-routing of passengers has been tackled in [7].

In railway transportation an important issue concerns the limited capacity of the track system. This has been taken into account, see [16] for modeling issues and [14, 13] for an integer programming formulation and heuristic approaches solving capacitated DM problems. 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. A similar approach has been used in [3], where capacity constraints for tracks and stations have been modelled in an alternative graph. In this paper, we additionally consider the possibility of re-optimizating the assignment of trains to platforms in the stations.

Our first example shows, that it is important to take station capacities into account. As a station only offers a given number of platforms, its capacity is limited. Ignoring the station capacity leads to solutions that might not be feasible in practice since it is implicitly assumed that infinitely many trains can wait in a station until there is room on the tracks such that they can continue their journeys.

▶ **Example 1.** Assume a busy piece of track consisting of stations S1, S2 and S3 along which every 10 minutes a train is running, and no shorter interval than 10 minutes between two such trains is allowed. The original schedule can be read off in the following table where the planned departure time in S1, the planned arrival time in S2, the planned departure time in S2 and the planned arrival time in S3 are given for 5 trains.

station	S1	S2			S3
	dep	arr	dep (planned)	*dep (delayed)*	arr (planned)
train 1	00	**15**	17	*17*	32
train 2	10	**25**	27	*57*	42
train 3	20	**35**	37	*67*	52
train 4	30	**45**	47	*77*	62
train 5	40	55	57	*87*	72

Now assume that train 1 gains a delay of say 30 minutes due to technical problems directly after leaving station S2. Without taking station capacities into account all trains following this delayed one would wait in S2 until the track is free again and would then leave one after another as can be seen in the column *dep (delayed)* in the above table. This means that train 2, 3, and 4 need to wait in the station simultaneously, since they all arrive before the track is freed. However, if there is only capacity for two trains in station S2, only trains 2 and 3 can enter station S2. Train 4 can therefore not enter the station at its planned time 45, but has to wait until either train 2 or train 3 has departed from station S2. This means that train 4 will not arrive before 57 and thus arrives at station S2 with a delay. This arrival

delay (which would be ignored if the station capacity is not taken into account) may even force train 4 to stay longer in station S1 and hence effect other trains at earlier stations.

Note that the problem of taking station capacities into account is also relevant in timetabling. Here one has to check for a given timetable if 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 pathing problem*, *i.e.*, to find routes through the stations for any of the trains using the detailed track topology. This feasibility problem has been extensively studied. In [9] 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. Another line of research aiming at real-time solutions is based on the alternative graph formulation [12], 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 [5, 1]. In [4], 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 problem has been modeled using a set packing approach in [11]. In [2] the problem is modelled as an ILP using clique inequalities in a conflict graph. For a recent survey on railway track allocation problems, see [10].

2 Integer programming formulation

In this section we will present two different integer programming formulations that take the capacities within stations into account. As basis for both models we will use the integer programming formulation which describes the delay management problem including capacities of the tracks as it was introduced in [14]. Note that other formulations of the DM problem can analogously be extended to take the stations' capacities into account.

For modelling DM problems as integer programs usually an *event-activity network* $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ is used as underlying directed graph. Its set of nodes $\mathcal{E}$ corresponds to all arrival and departure events of all trains at all stations. 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 *changing activities* $\mathcal{A}_{\text{change}}$ linking an arrival of a train in a station to a (later) departure of another train in the same station. Finally, *headway activities* $\mathcal{A}_{\text{head}}$ are needed for any pair of trains competing for the same infrastructure after their departures. We will denote the minimal duration of an activity a as L_a.

The most important decision is which connections need to be kept alive. For each changing activity $a \in \mathcal{A}_{\text{change}}$ 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 $(i, j) \in \mathcal{A}_{\text{head}}$ 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 variables $x = (x_i)$ therefore define 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.

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

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

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

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

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

$$g_{ij} + g_{ji} = 1 \qquad \forall (i,j) \in \mathcal{A}_{\text{head}}, \tag{6}$$

$$x_i \in \mathbb{N} \qquad \forall i \in \mathcal{E}, \tag{7}$$

$$z_a \in \{0,1\} \qquad \forall a \in \mathcal{A}_{\text{change}}, \tag{8}$$

$$g_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}_{\text{head}}. \tag{9}$$

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. Also here we weight the changing activity a with the number of passengers c_a who planned to use it as a transfer. The objective is an approximation for the overall delay of all passengers and rather commonly used in DM. It gives the exact value if the never-meet property for headways holds (see [14]). A more realistic model taking into account the real paths passengers would use in case of delays has been developed in [7]. It can also be used as basis for our extension, but is technically more difficult and computationally harder to solve. The interpretation of the constraints is as follows: (2) makes sure that no train departs 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 events competing for the same infrastructure (6) makes sure that exactly one of the two headway constraints is respected and (5) propagates the delay along this headway activity.

2.1 A packing-based integer programming formulation

In order to take the limited capacity of the stations into account, the first integer programming approach counts the number of trains in a station at a certain time and restricts this number by the number of platforms C_s. To this end, let τ be the largest possible time an event can take place in a reasonable timetable. We introduce binary variables y_{it} for all events $i \in \mathcal{E}$ and times $t = 1, \ldots, \tau$, that are defined as

$$y_{it} = \begin{cases} 1 & \text{if event } i \text{ takes place before or at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

The following constraints ensure that y_{it} takes the correct value for all events i and times t where M is a sufficiently large number, *e.g.*, $M \geq \tau + 1$.

$$y_{it} \geq \frac{t - x_i + 1}{M} \quad \text{and} \quad 1 - y_{it} \geq \frac{x_i - t}{M}. \tag{10}$$

For $x_i \leq t$, the left equation forces y_{it} to 1, while the right constraint is redundant. On the other hand, for $x_i > t$, the right constraint forces y_{it} to zero while the left constraint is redundant.

In order to limit the number of trains at station s at time t, we now count the number of trains that are present at the station for each time t. 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 the train starts to enter the station will be 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 the last car of the train leaves the platform and hence the time the next train can start to enter. Thus $[h_i, h_{i'}]$ denotes the interval during which a platform is occupied. 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. When counting the number of trains, we should not consider the time x_i that the arrival event i takes place, but the time h_i that the train starts using the platform. Observing that $h_i \leq t \Leftrightarrow x_i \leq t + l_i$, this can be done by shifting the y variables. A similar remark holds for departure events. This leads to the following constraints, that limit the number of trains in the stations.

$$\sum_{i \in \mathcal{E}^s_{\text{arr}}} y_{i(t+l_i)} - \sum_{i \in \mathcal{E}^s_{\text{dep}}} y_{i(t-l_i)} \leq C_s \quad \forall s \in S, t \in \{1, \ldots, \tau\}, \tag{11}$$

where $\mathcal{E}^s_{\text{arr}}$ and $\mathcal{E}^s_{\text{dep}}$ denote the set of arrival and departure events at station s, respectively and C_s represents the number of platforms in station s.

Adding the new constraints (10),(11), and $y_{it} \in \{0, 1\}$ for all $i \in \mathcal{E}, t \in \{1, \ldots, \tau\}$ to the formulation (1)-(9) we obtain our first integer programming formulation (DM-Cap-1) for the DM problem with capacity constraints.

2.2 An assignment-based integer programming formulation

The second integer programming formulation views a station as a set of platforms, and introduces headway constraints for trains that make use of the same platform. 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 the set P_s of platforms at station $s \in S$. 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

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

Of course, each arrival event must be assigned to exactly one platform.

$$\sum_{p \in P_s} y_{ip} = 1, \quad \forall s \in S, i \in \mathcal{E}^s_{\text{arr}}. \tag{12}$$

In order to model the limited capacity of the stations, we determine the order in which the trains arrive at a certain platform. Consider two trains t_1 and 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, 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

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

If the trains are assigned to the same platform, either t_1 must have departed before t_2 arrives, or t_2 must have departed before t_1 arrives. Denoting $a_i = (i, i')$ as the waiting activity of train t_1 and $a_j = (j, j')$ as the waiting activity of train t_2 this is modelled by the following set of constraints.

$$x_j - x_{i'} + M\bar{g}_{ij} \geq L_{ij} = l_{i'} + l_j, \tag{13}$$
$$x_i - x_{j'} + M\bar{g}_{ji} \geq L_{ji} = l_{j'} + l_i, \tag{14}$$
$$\bar{g}_{ij} + \bar{g}_{ji} \leq 3 - y_{ip} - y_{jp} \quad \forall p. \tag{15}$$

l_i is defined as in Section 2.1, hence L_{ij} describes the time during which the platform is occupied after the departure of i' and before the arrival of train j (*i.e.*, when it opens its doors). 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. 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. On the other hand, if trains t_1 and t_2 are assigned to the same platform 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 must be satisfied.

The above constraints must be introduced for each pair of trains t_1, t_2 that dwell at a common station $s \in S$. Note that this type of constraints has also been used to model alternative graphs (see [12]).

Adding the constraints (12)-(15), and $y_{ip} \in \{0, 1\}$ for all stations $s \in S$ and $i \in \mathcal{E}^s_{\text{arr}}, p \in P_s$ to the formulation (1)-(9) we obtain our second integer programming formulation (DM-Cap-2) for the DM problem with capacity constraints. Note that this formulation reduces to a problem of type (DM) if the assignment of events to platforms is determined in advance.

▶ **Lemma 2.** *For fixed variables y_{ip} for all $i \in \mathcal{E}^s_{arr}, p \in P_s$ the formulation (DM-Cap-2) reduces to an instance of (DM),* i.e.*, can be solved as DM problem with headway constraints.*

Proof. If all y_{ip} variables are fixed we have two possibilities for (15): Either both y_{ip} variables are 1, then $\bar{g}_{ij} + \bar{g}_{ji} \leq 1$ and (13)-(14) reduce to a headway constraint of type (5)-(6), or at least one of the y_{ip} variables is 0, then (13)-(15) becomes redundant. ◀

Note that for a fixed assignment of trains to platforms this result can be interpreted as if we introduced a track for each platform within the stations. This give rise to the following two bounds which can easily be calculated using an algorithm that solves problem (DM).

First, it is clear that (DM) is a relaxation of (DM-Cap-2) (and of (DM-Cap-1)), hence its objective value z^{DM} is a lower bound. On the other hand, if we fix the assignment y of trains to stations in (DM-Cap-2) we obtain an upper bound $z^*(y)$ which can also be calculated by any algorithm for (DM) according to Lemma 2. Hence we can compute an upper and a lower bound, *i.e.*, $z^{DM} \leq z^* \leq z^*(y)$. We will denote the model with a fixed platform assignment by DM-Fix.

2.3 Computational results

We have performed a computational test to see which of the above formulations performs best. Our test considers the railway network in the Randstad, which is the mid-Western part of the Netherlands, where the railway network is very dense. We have created two cases, that contain all long distance trains on this network during a period in the evening. For each case, we generated 100 delay scenarios and solved the corresponding DM problem with both formulations. Table 1 gives an impression of the sizes of the instances and of the resulting

			Size of the program			(DM-Cap-1)		(DM-Cap-2)	
Case	Stations	Trains	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}_{\text{head}}\rvert$	$\lvert\mathcal{A}_{\text{plat}}\rvert$	Bin.	Con.	Bin	Con
I	10	117	344	623	3836	166323	172048	9927	23492
II	16	168	576	986	6265	266608	810188	16316	38457

Table 1 Some characteristics of the case and the resulting integer programs. $\mathcal{A}_{\text{plat}}$ denotes the set of train pairs (t_1, t_2) that dwell at a common station. Bin. and Con. give the number of binary variables and constraints in the integer program, respectively.

Formulation		Case I		Case II	
		Obj. Value	Time (s)	Obj. Value	Time (s)
DM	Neglecting capacity	248210	0.42	888908	0.65
DM-Cap-1	Packing-based	277959	781.9	-	-
DM-Cap-2	Assignment-based	277959	9.95	1013300	54.46
DM-Fix	Fixed platforms	330415	1.76	1146420	5.27

Table 2 The objective values and solution times for the various formulations.

integer programs for both formulations. It can be observed from the table that the second formulation requires less variables and constraints than the first one. This suggests that the second formulation will solve the problem much faster.

We have used Cplex 12.2 on an Intel Core i5-2410M with 4 GB of RAM to solve the integer programs. We set the maximal running time of the algorithm to 20 minutes. As objective value for a formulation, we take the average objective value over all 100 delay scenarios. Table 2 shows these objective values and the solution times for both formulations. For comparison, we also included the objective value and solution time of the model that neglects the limited station capacity. We see in the table that the objective value increases if we explicitly model the limited capacity of the stations. This implies that the model that ignores the station capacity finds a solution that is infeasible in practice. For Case I, we see that the second formulation is much faster than the first one. For Case II, Cplex could not solve all instances with the first formulation within the available computation time. Only in 63 instances, the optimal solution is found. In 15 instances, a feasible solution was found but not a provably optimal one. Finally, in the remaining 22 instances no solution was found at all. These results are in line with what can be expected based on the number of binary variables, which is much smaller for the second formulation. Finally, if the platform assignment is fixed as in the timetable, worse solutions are found. This shows that it pays off to schedule the trains in a station dynamically.

3 An iterative approach

The integer programming formulation (DM-Cap-2) yielded a big improvement concerning the running time. Still, for large instances, making wait-depart-decisions, priority decisions and platform assignments simultaneously is intractable. We thus 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 [14]. 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-2) we obtain:

1. Fix the station assignment y_{ip} in (DM-Cap-2) according to the planned timetable.
2. Solve the resulting problem (DM-Cap-2) with fixed y_{ip} and obtain solution with disposition timetable x_i, wait depart decisions z_a and priority decisions g_{ij} and $\bar{g}_{ij}$
3. For every station find a new 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 big instances of DM decomposing the problem into two steps still results in long running times, we can use the approach of [13] to decompose Step 2 of the algorithm further into two smaller subproblems making first the priority decisions and the wait-depart decisions afterwards.

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 DM step to shift events forward in time, if possible.

In the following we will 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 will hopefully yield a better disposition timetable in the next iteration of Step 2. Recall from (13) and (14) 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' or j, j', respectively, that are assigned to the same platform.

We process every 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 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. 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 will be occupying the station during the time interval $(h_i, h_{i'})$. If a train is delayed, we distinguish two cases:

- There is another train which occupies the interval $(h_j, h_{j'})$ with $h_{j'} = h_i$ and which is on the same platform p, *i.e.*, $y_{ip} = y_{jp} = 1$. In this case, a new assignment might help to reduce the delay of i. Assuming that $(k, i) \in \mathcal{A}_{\text{drive}}$ is the preceding driving activity of the train we define the *wish time* of i as $w_i := x_k + L_{ki} + d_{ki} - l_i$.
- If no other train is on the same platform 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* $P_s = \{1, \dots, P\}$ *and for every arrival event* $i \in \mathcal{E}^s_{arr}$ *an interval* $[h_i, h_{i'}]$ *and a wish time* $w_i \le h_i$ *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 assignment* y_{ip} *such that for all* $i, j \in \mathcal{E}^s_{arr}$ *and* $p \in P_s$

$$q_j \in (q_i, h_{i'}) \Longrightarrow y_{ip} + y_{jp} \le 1 \tag{16}$$

and $\sum_{i \in \mathcal{E}^s_{arr}} c_i q_i$ *is minimal.*

Note that $q_j \in (h_i, h_{i'})$ or $q_i \in (h_j, h_{j'}) \Longleftrightarrow (q_i, h_{i'}) \cap (q_j, h_{j'}) \neq \emptyset$, *i.e.*, if and only if the two trains belonging to i and j cannot be scheduled on the same platform. This problem can be formulated as mixed-integer program as it is but the formulation does not seem to be promising due to condition (16). Instead we will show that (PA) is polynomially solvable by first identifying a finite dominating set $\mathcal{C}$ for the q_i variables. We then notice that for every choice of the q_i variables, we can check feasibility by solving a coloring problem. Since checking all possible $q \in \mathcal{C}^{|\mathcal{E}^s_{\text{arr}}|}$ would lead to an exponential number of coloring problems, we will use that the considered graph is an interval graph and code the solvavability of the coloring problem in the constraints of an IP formulation for which we are able to show that its coefficient matrix is totally unimodular. Our first result concerns the finite dominating set.

▶ **Lemma 3.** *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 for which $y_{ip} = 1$, $q_i \ge \max\{h_{j'} : y_{j'p} = 1 \text{ and } h_{j'} \le q_i\}$. Now assume that $q_i \notin \mathcal{C}$ for some $i \in \mathcal{E}^s_{\text{arr}}$. Let p the platform with $y_{ip} = 1$. Define

$$\tilde{q}_i := \max\{w_i, \max\{h_{j'} : y_{j'p} = 1 \text{ and } h_{j'} \le q_i\}\}. \tag{17}$$

Then $\tilde{q}_i \in [w_i, h_i]$ and for all j condition (16) 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 we have to check if there is a platform assignment y such that (16) is satisfied. To this end we transform our problem into a coloring problem in the following graph $G(q) = (\mathcal{E}^s_{\text{arr}}, E)$: For every $i \in \mathcal{E}^s_{\text{arr}}$ we draw 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. 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. [18, Chapter 65]). 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 $\mathcal{C} = \{q^1, \dots, q^{|\mathcal{C}|}\}$ in increasing order and let us define intervals $I_k := (q^k, q^{k+1})$ for $k = 1, \dots |\mathcal{C}| - 1$. For a given q we define a matrix $A(q) = (a_{il})$ with $|\mathcal{E}^s_{\text{arr}}|$ rows and $|\mathcal{C}| - 1$ columns and entries

$$a_{il} = \begin{cases} 1 & \text{if } (q_i, h_{i'}) \cap I_l \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

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

▶ **Lemma 4.**

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

Proof. Due to Lemma 3 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 $a_{il} = a_{jl} = 1$. 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 $a_{il} = 1$ 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 4 in the constraints. Denote by q_i^k the entries of the set $\mathcal{C}_i = \{q_i^1, q_i^2, \dots, q_i^{|\mathcal{C}_i|}\}$. Then for every arrival event i and every choice $q_i^k \in \mathcal{C}_i$ we define the variables:

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

These will be 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 stand together. We need to extend the matrix defined in (18) to all possible choices of q. To this end for every $q_i^k \in \mathcal{C}_i$ we define a row with

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

Doing this for all $i = 1, \dots, |\mathcal{E}^s_{\text{arr}}|$ we obtain a matrix $\tilde{A} = (\tilde{a}_{il}^k)$ with $\sum_{i \in \mathcal{E}^s_{\text{arr}}} |\mathcal{C}_i|$ rows and $|\mathcal{C}| - 1$ 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) rows in $\tilde{A}$. (PA) can hence be rewritten as

$$\min \sum_{i \in \mathcal{E}^s_{\text{arr}}} c_i \sum_{k=1}^{|\mathcal{C}_i|} q_i^k \eta_i^k \tag{19}$$

$$\text{such that } \sum_{k=1}^{|\mathcal{C}_i|} \eta_i^k = 1 \qquad \text{for all } i \in \mathcal{E}^s_{\text{arr}}, \tag{20}$$

$$\sum_{i \in \mathcal{E}^s_{\text{arr}}} \sum_{k=1}^{|\mathcal{C}_i|} \tilde{a}_{il}^k \eta_i^k \leq P \qquad \text{for all } l \in 1, \dots, |\mathcal{C}| - 1, \tag{21}$$

$$\eta_i^k \in \{0, 1\} \qquad \text{for all } \forall i \in \mathcal{E}^s_{\text{arr}},\ \forall k \in \mathcal{C}_i. \tag{22}$$

▶ **Lemma 5.** *The constraint matrix A' defined by the inequalities (20)-(21) is totally unimodular.*

Proof. We will show that A' is totally unimodular by showing that every subset J of rows of A' can be partitioned into two sets J_1, J_2 with $J_1 \cap J_2 = \emptyset$, $J_1 \cup J_2 = J$ and $\sum_{j \in J_1} a'_{jl} - \sum_{j \in J_2} a'_{jl} \in \{-1, 0, 1\}$ for all columns l (see for example [18, Chapter 5]). The columns of A' are associated to the variables of our integer program. For every $i = 1, \dots, \mathcal{E}^s_{\text{arr}}$ we will denote by $C(i)$ the indices of the columns of A' associated to a variable η_i^k. The rows represent the constraints. For the first rows $i = 1, \dots, |\mathcal{E}^s_{\text{arr}}|$ we thus have

$$a'_{il} = \begin{cases} 1 & \text{if the column } l \text{ belongs to variable } \eta_i^k \text{ for a } k, \\ 0 & \text{otherwise.} \end{cases}$$

Starting from row $|\mathcal{E}^s_{\text{arr}}| + 1$, the matrix A' consists of the matrix $\tilde{A}^T$. We notice that $\tilde{A}$ has the consecutive ones property and that for a given $i = 1, \ldots, |\mathcal{C}_i|$, all columns of $\tilde{A}^T$ with index in $C(i)$ have their last 1-entry in the row that represents the constraint for the interval with end point $h_{i'}$.

Let J be an index set of rows of A' and $J^A = J \setminus \{1, \ldots, |\mathcal{E}^s_{\text{arr}}|\}$, that is the part of the chosen subsets that is contained in $\tilde{A}^T$. For each subset J of rows, we now define

$$S(J, l) = \sum_{j \in J} a'_{jl}.$$

We then alternately assign the rows J^A to two sets J^A_1 and J^A_2. Then for every i and every column associated to a variable η^k_i either

$$S(J^A_1, l) - S(J^A_2, l) \in \{0, 1\} \quad \text{or} \quad S(J^A_1, l) - S(J^A_2, l) \in \{-1, 0\}, \tag{23}$$

because of the consecutive ones property and because for every i the last 1-entry of $C(i)$ is in the same row. We set $J_1 := J^A_1$ and $J_2 := J^A_2$ and add the indices of the first $|\mathcal{E}^s_{\text{arr}}|$ rows in the following way to these sets: If for row i the left inclusion in (23) holds, we assign the i-th row to J_2, if the right inclusion holds, we assign it to J_1. We obtain

$$S(J_1, l) - S(J_2, l) \in \{-1, 0\} \quad \text{or} \quad S(J_1, l) - S(J_2, l) \in \{0, 1\},$$

respectively. This proves total unimodularity. ◀

▶ **Corollary 6.** *(PA) can be solved by linear programming.*

4 Conclusion and further research

In this paper, we introduced a DM model that incorporates the limited capacity of railway stations. We have given two approaches that can be used to extend any integer programming formulation for the DM problem. Our first approach determines the number of trains in a station at each time and requires this number to be smaller than the station capacity. The second approach views a station as a set of parallel tracks and determines an assignment of trains to platforms explicitly. In a computational test, the second formulation strongly outperforms the first one.

As solutions to the DM models should be available within a very short computation time, we also proposed an iterative solution method for the DM model with station capacities. This heuristic iterates between solving the DM problem with a given platform assignment and optimizing the platform assignment given the timetable and wait-depart decisions. We show that determining an improving platform assignment can be done in polynomial time. Two main directions for further research should be considered. First, the second integer program should be tested on larger real-world instances and the performance of the iterative heuristic should be evaluated. Also other heuristics, e.g., exchange heuristics, could be implemented and compared. Second, after assigning platforms to the trains, a route through the station has to be determined for each train. Solving the DM and routing problem simultaneously might be computationally intractable. However, we plan to integrate the routing decisions in the platform assignment step of the iterative heuristic.

References

1 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.

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

3 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*, In Press, 2010.

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.

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

6 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.

7 T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay management with re-routing of passengers. *Transportation Science*, 2011. to appear.

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

9 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.

10 R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR Spectrum*, pages 1–41, 2009. in press.

11 R. Lusby, J. Larsen, D. Ryan, and M. Ehrgott. Routing trains through railway junctions: A new set packing approach. *Transportation Science*, 45(2):228–245, 2011.

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

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

14 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.

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

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

17 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.

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