Delay Management with Re-Routing of Passengers

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] Department of Logistics, 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 EI 2010-31

April 2010

Abstract

The question of delay management is whether trains should wait for a delayed feeder train or should depart on time. In classical delay management models passengers always take their originally planned route. In this paper, we propose a model where re-routing of passengers is incorporated.

To describe the problem we represent it as an event-activity network similar to the one used in classical delay management, with some additional events to incorporate origin and destination of the passengers. We present an integer programming formulation of this problem. Furthermore, we discuss the variant in which we assume fixed costs for maintaining connections and we present a polynomial algorithm for the special case of only one origin-destination pair. Finally, computational experiments based on real-world data from Netherlands Railways show that significant improvements can be obtained by taking the re-routing of passengers into account in the model.

Keywords: Public Transportation, Delay Management, Re-Routing, OD-pairs

*Corresponding author

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. To ensure a high frequency and an easy to remember timetable, most European railway companies have opted for a cyclic timetable (see Liebchen (2008) or Kroon et al. (2009) for two recent publications on the subject). In such a timetable, each line has to be operated in a cyclic, or periodic, pattern: the trains run, for example, every 30, 60 or 120 minutes. A weak point in such a system is that passengers often have to change trains, since it is impossible to give a direct connection between all origin-destination pairs. 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 preferably with a cross-platform connection, i.e. both trains stop at two adjacent tracks of the same platform. However, if train A has a delay during the operations, the question is whether train B should wait or depart. Such decisions are called *delay management*.

Delay management 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*. Since delays are often transferred if a connecting train waits for a delayed feeder train such connections are often not maintained in case of delays.

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 (*wait-depart decisions*). A first integer programming formulation for this problem has been given in Schöbel (2001) and has been further developed in 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) where it turns out that the problem is NP-hard even in very special cases. The online version of the problem has been studied in Gatto et al. (2007) and Gatto (2007). In Berger et al. (2007), it was shown that the online version of the delay management problem is PSPACE-hard. Further publications about delay management include a model in the context of max-plus-algebra (de Vries et al., 1998; Goverde, 1998), a formulation as discrete time-cost tradeoff problem (Ginkel and Schöbel, 2007) and simulation approaches (Suhl and Mellouli, 1999; Suhl et al., 2001). Recently, also the limited capacity of the track system has been taken into account, see Schöbel (2009) for modeling issues and Schachtebeck and Schöbel (2010); Schachtebeck (2010) for heuristic approaches solving capacitated delay management problems.

What has been neglected so far is the aspect of re-routing. In the available models it is assumed that passengers take exactly the lines they planned, i.e. if they miss a connection they have to wait a complete period of time (the cycle time) until the same connection takes place again.

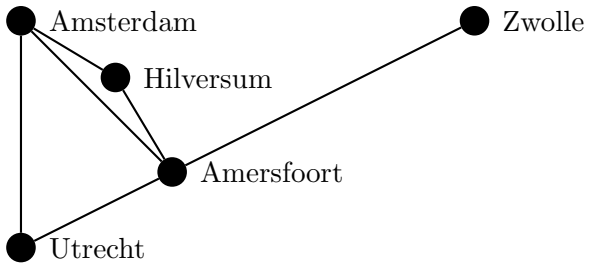

Figure 1: A small part of the railway network in the Netherlands. A regional train runs from Amersfoort to Hilversum and further to Amsterdam. An intercity service runs from Zwolle to Utrecht and stops at station Amersfoort. All other trains are intercities as well.

This assumption is usually not valid in practice. Often there is an earlier connection using another line or even changing the path of the trip. A real-world example of a situation where re-routing passengers in case of delays is beneficial, is given next.
Consider the network in Figure 1. An intercity service runs from Zwolle to Utrecht via Amersfoort. There are also intercities from Utrecht to Amsterdam and from Amersfoort to Amsterdam. Finally, a regional train runs from Amersfoort via Hilversum to Amsterdam. A large number of passengers want to travel from Zwolle to Amsterdam, and thus have a transfer at Amersfoort. In the current timetable, the intercity to Amsterdam departs from Amersfoort 5 minutes after the intercity from Zwolle has arrived. Therefore, if the intercity from Zwolle has a small delay, these passengers will miss the connecting intercity to Amsterdam. If the possibility of re-routing the passengers is not taken into account, the decision to delay the intercity from Amersfoort to Amsterdam assumes that the passengers that miss the connection at Amersfoort have to wait for one hour for the next intercity. However, if the connection is missed, the passengers can also stay in the delayed train and transfer in Utrecht instead. The transfer time in Utrecht is larger than in Amersfoort. If the delay of the train in Amersfoort is too large for the transfer in Utrecht, the passengers should better transfer to the regional train in Amersfoort. Although the travel time of the regional train is larger than that of the intercity, the total delay of the passengers will be higher if they wait for the next intercity from Amersfoort to Amsterdam. This small example shows that the delay of passengers that miss a connection is often much smaller than the cycle time of the timetable. To find optimal wait-depart decisions, re-routing passengers should therefore be taken into account.
In this paper we will investigate how such a re-routing of passengers can be incorporated into the delay management model. We denote the resulting model by *delay management with re-routing decisions (DMwRR)*. To the best of our knowledge a re-routing of passengers has never been treated before. The contributions of our paper are as follows. Firstly, we have developed a new model and integer programming formulation for DMwRR. Secondly, we prove that DMwRR is NP-hard in the general case, while it is polynomially solvable if

only one origin-destination pair is present. And finally, our third contribution is that we show that DMwRR can be solved for medium-size real-world instances and performs significantly better than the existing models where re-routing is not taken into account.
The remainder of the paper is structured as follows. In Section 2 we show how the re-routing of passengers can be modeled in the event-activity network. An integer program using event-activity networks is formulated in Section 3. In Section 4 we present a polynomially solvable special case of the problem. We show that a slight generalization of this case is already NP-hard. Furthermore, we discuss another simplified variant in which we assume fixed delay costs for each maintained changing activity. In Section 5, we report the results of several experiments based on real-world data of Netherlands Railways, the largest passenger operator on the Dutch railway network. Finally, we conclude the paper mentioning ideas for further research.

2 Model

We assume that the number of passengers that want to travel from a given origin to a destination at a certain time is known. For example, 200 passengers want to travel from Zwolle to Amsterdam at 8 o'clock in the morning. We denote such an origin-destination pair by $p = \{u, v, s_{uv}\}$, where u is the origin, v is the destination and s_{uv} is the planned starting time of the trip. $\mathcal{P}$ denotes the set of all such origin-destination pairs. From now on, we will abbreviate an origin-destination pair as an OD-pair. We use w_p for the number of passengers associated to an OD-pair $p \in \mathcal{P}$.
Given some source delays from outside the system, the delay management problem is to decide which trains should wait for delayed feeder trains and which trains should depart on time. The goal is to find a solution which is best for the passengers. In our work we want to minimize the sum of all delays over all OD-pairs assuming that all passengers take shortest paths.
In order to model the delay management problem with re-routing we will make use of event-activity networks, first introduced by Nachtigall (1998) for timetabling problems and used for the classical delay management problems by Schöbel (2006). The event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ is a directed graph, where $\mathcal{E}$ denotes the set of events and the set $\mathcal{A}$ consists of the activities. The departure or the arrival of a train g at a station v, denoted by $(g - v - Dep)$ or $(g - v - Arr)$ respectively, are the most important events in the network. To incorporate the routes of the passengers, we introduce for every OD-pair $p = \{u, v, s_{uv}\} \in \mathcal{P}$ an origin event $(p - Org)$ and a destination event $(p - Dest)$. Note that besides the origin and the destination, the OD-pairs also contain the time at which passengers want to start their journeys. In summary, the set of events in the network, denoted by $\mathcal{E}$, consists of the departure events of the trains, the arrival events of the trains and the origin and destination events for

the passengers for a given OD-pair.

$$\mathcal{E} = \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{org}} \cup \mathcal{E}_{\text{dest}}.$$

The activities are the arcs in the directed graph $\mathcal{N}$. There are driving arcs, waiting arcs and changing arcs. The driving and waiting arcs represent driving from one station to the next and waiting at a station to let the passengers get on and off the train. The changing activities are used by the passengers. They represent the possibility for passengers to transfer from a train that arrives at a certain station to a train that departs at the same station some time later. It should be noted that the driving and waiting arcs impose operational restrictions on the vehicles. On the contrary, a changing arc does not imply that a train has to wait in case of a delay of another train, although it would be convenient for the transferring passengers. In fact, the decision "wait or depart on time" is the main decision we want to take during the optimization process.

To take the re-routing of passengers into account, we additionally introduce origin and destination arcs. Let an origin event $e = (p - Org) \in \mathcal{E}_{\text{org}}$ be given, where $p = \{u, v, s_{uv}\}$ represents the passengers that want to travel from station u to station v at time s_{uv}. This event e is connected to all departure events that depart from u not earlier than the time s_{uv}. It remains to connect the arrival events to the destination events. Consider therefore a destination event $(p - Dest) \in \mathcal{E}_{\text{dest}}$, where again $p = \{u, v, s_{uv}\}$. Denote SP_p as the earliest arrival time of the passengers if there are no delays and denote n_p as the number of transfers needed for this trip. SP_p is clearly a lower bound on the arrival time of the passengers. To derive an upper bound on the arrival time, note that in the worst case all n_p connections are missed (and no other route is possible). Let d_e denote the source delay occurring at an event $e \in \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}}$. As the overall delay at a node cannot exceed $\max_{e' \in \mathcal{E}} d_{e'}$, an arrival event e should be connected to $(p - Dest)$ if e is an arrival event at station v and if the planned time π_e satisfies $\pi_e \in [SP_p, SP_p + n_p T + \max_{e' \in \mathcal{E}} d_{e'}]$, where T is the cycle time of the original timetable and $\max_{e' \in \mathcal{E}} d_{e'} \leq T$. This concludes the description of the arcs in the event activity network. Summarizing,

$$\mathcal{A} = \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{change}} \cup \mathcal{A}_{\text{org}} \cup \mathcal{A}_{\text{dest}}.$$

An example of an event-activity network is given in Figure 2. This event-activity network corresponds to the railway network in Figure 1. The oval nodes represent the origin and destination events, that are introduced to model the behavior of passengers when delays occur. The dashed arcs depicting the origin and destination arcs are needed to take routing of passengers into account.

For every activity $a \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{change}}$ a length L_a is given that represents the technically minimal time that is needed to perform the activity. As the origin and destination activities are not activities in the original sense and thus they are not time consuming, their lengths

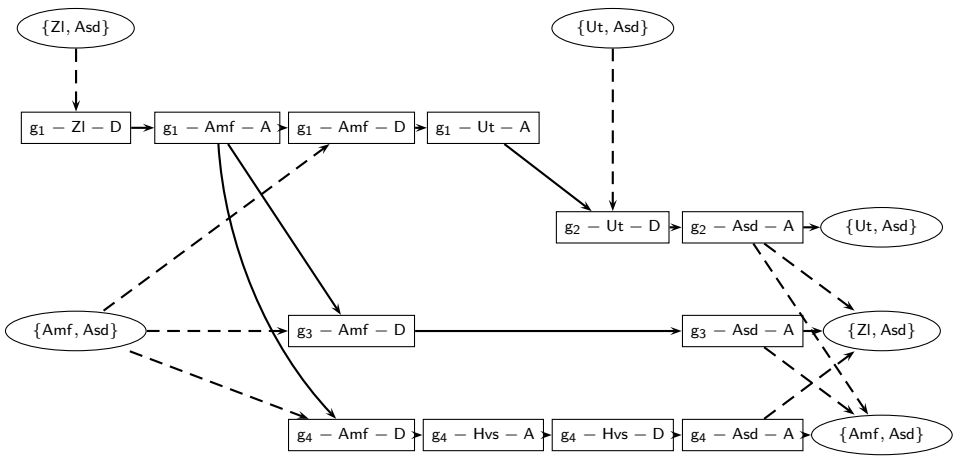

Figure 2: The event activity network for the situation depicted in Figure 1. The square nodes are the departure and arrival events where "D" stands for departure and "A" stands for arrival. The origin and destination events are represented by ovals omitting the add-ons "Org" or 'Dest" as this is obvious in the picture. As we only consider one possible departure time for each origin-destination pair, we did not include the starting time in the origin and destination nodes. The dashed arcs are the origin and destination arcs, that are introduced to be able to state the shortest path problem for the passengers. The solid lines represent driving, waiting and changing activities.

can be set to 0.

For every event $e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$, the planned time is denoted by π_e, i.e. π corresponds to the timetable as it is planned to be operated. For an origin event $e = (p - Org) \in \mathcal{E}_{\text{org}}$ with $p = \{u, v, s_{uv}\}$ we set $\pi_e = s_{uv}$ (which can be interpreted as the time at which a passenger of OD-pair p arrives at his or her departure station). For destination events we have to determine the time when the passengers reach their last station, hence π_e is not known beforehand.

Given a timetable, for every OD-pair a route through the network has to be found, so that the travel time is minimized. To this end, let P be a directed path from e_1 to e_2 in the network $\mathcal{N}$. We now define its length $l(P)$.

- First, assume that $e_2 \in \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}}$. We define $l(P) = \pi_{e_2} - \pi_{e_1}$ to be the travel time or distance between e_1, e_2 in $\mathcal{N}$.
- We now extend this definition to nodes $e_2 \in \mathcal{E}_{\text{dest}}$. Let $pre(e_2, P)$ be the predecessor of e_2 in path P from e_1 to e_2. Then we define $l(P) = \pi_{pre(e_2,P)} - \pi_{e_1}$.
- We are mainly interested in the travel time for the passengers. For the special case of a path P connecting an OD-pair $p = \{u, v, s_{uv}\}$ we hence obtain $l(P) = \pi_{pre(e_2,P)} - s_{uv}$. As we assume that passengers take the fastest paths to arrive at their destinations, we

set $l(p) = l(\hat{P}_{uvs_{uv}})$ where $\hat{P}_{uvs_{uv}}$ is a fastest path from the origin event $e = (p - Org)$ to the destination event $e = (p - Dest)$.

Given a set of source delays d_e associated to some events $e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$ the problem is to decide which trains should wait for passengers to arrive from delayed trains and which should depart without waiting. Thus we have to determine which of the connections $a \in \mathcal{A}_{\text{change}}$ will be maintained and which will be removed. We denote the set of maintained connections by $\mathcal{A}_{\text{fix}}$. For the resulting network

$$\mathcal{N}(\mathcal{A}_{\text{fix}}) := (\mathcal{E}, \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{fix}} \cup \mathcal{A}_{\text{org}} \cup \mathcal{A}_{\text{dest}})$$

in which the set of changing arcs has been replaced by $\mathcal{A}_{\text{fix}}$ a new timetable can be constructed using the critical path method (see Schöbel (2007)). The times for the events $e \in \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}}$ in this new timetable will be denoted by x_e. For an OD-pair p we define $t_{\mathcal{A}_{\text{fix}}}(p) = x_e$ where e is the predecessor of the destination event $(p - Dest)$ on a shortest path from the origin event $(p - Org)$ to the destination event $(p - Dest)$ in the network $\mathcal{N}(\mathcal{A}_{\text{fix}})$.
In $\mathcal{N}(\mathcal{A}_{\text{fix}})$ the travel time of an OD-pair $p = \{u, v, s_{uv}\}$ is analogously defined as

$$l_{\mathcal{A}_{\text{fix}}}(p) = t_{\mathcal{A}_{\text{fix}}}(p) - s_{uv}.$$

In the delay management problem we want to minimize the sum of all delays of the OD-pairs. The delay of an OD-pair $p = \{u, v, s_{uv}\}$ is given as

$$l_{\mathcal{A}_{\text{fix}}}(p) - l(p) = t_{\mathcal{A}_{\text{fix}}}(p) - s_{uv} - l(p).$$

Summarizing, the objective of delay management with re-routing is to find a subset $\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}$ so that we minimize:

$$\min_{\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}} \sum_{p \in \mathcal{P}} w_p \cdot (t_{\mathcal{A}_{\text{fix}}}(p) - SP_p) \quad \text{or, equivalently} \quad \min_{\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}} \sum_{p \in \mathcal{P}} w_p \cdot t_{\mathcal{A}_{\text{fix}}}(p).$$

In words: we minimize the average delay or the sum of the arrival times of the passengers. Since delay management without re-routing is NP-hard (Gatto et al., 2005), it is not surprising that delay management with re-routing is NP-hard as well. In Section 4 we will investigate the borderline between NP-hardness and tractability by analyzing the complexity of delay management with re-routing for different structures of the underlying OD-data.

3 Integer Programming Formulation

In this section we will give an integer programming formulation that takes the routing decisions for the passengers into account explicitly. The model is based on the classical delay

management model as it was introduced in Schöbel (2007).
The event activity network is a directed graph. We denote $\delta^{\text{in}}(e)$ and $\delta^{\text{out}}(e)$ for the set of arcs into e and out of e, respectively, for every event $e \in \mathcal{E}$.

3.1 Variables

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.

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

The times that the arrival and departure events take place are the next set of decision variables. For each event $e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$, we define $x_e \in \mathbb{N}$ as the rescheduled time that event e takes place. The variables $x = (x_e)$ therefore define the disposition timetable. These decision variables are the same as in the classical model.
The new aspect that we have to model are the routes that the passengers take. First note that a route has to be determined for every origin-destination pair. Recall that $\mathcal{P}$ denotes the set of all origin-destination pairs. To model the routing decisions for a given pair $p \in \mathcal{P}$, we introduce binary decision variables q_{ap}, which indicate whether arc $a \in \mathcal{A}$ is used in the path that is chosen for origin-destination pair $p \in \mathcal{P}$. Formally, the variables q_{ap} are defined as follows.

$$q_{ap} = \begin{cases} 1 & \text{if connection } a \text{ is used by passengers in } p, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the arcs $a = ((p - Org), e) \in \mathcal{A}_{\text{org}}$ for $p \in \mathcal{P}, e \in \mathcal{E}_{\text{dep}}$ can only be used by the OD-pair p. Therefore, the variable $q_{ap'}$ for $a = (p - Org, e)$ and $p' \in \mathcal{P}$ is only needed if $p' = p$. A similar remark holds for q_{ap} with $a = (e, p - Dest) \in \mathcal{A}_{\text{dest}}$.
The arrival time for an OD-pair p now depends both on the path that is chosen, and on the disposition timetable x. To be able to incorporate the arrival time of these passengers in a linear model, we introduce a variable $t_p \in \mathbb{N}$, which will represent the arrival time for pair $p \in \mathcal{P}$.

3.2 Integer programming formulation

We first present our integer programming formulation for (DMwRR) and then discuss its meaning.

$$\min \sum_{p \in \mathcal{P}} w_p (t_p - SP_p) \tag{1}$$

such that

$$x_e \geq \pi_e + d_e \qquad \forall e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}, \tag{2}$$

$$x_e \geq x_{e'} + L_a \qquad \forall a = (e', e) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}, \tag{3}$$

$$x_e \geq x_{e'} + L_a - M_1(1 - z_a) \qquad \forall a = (e', e) \in \mathcal{A}_{\text{change}}, \tag{4}$$

$$q_{ap} \leq z_a \qquad \forall p \in \mathcal{P}, a \in \mathcal{A}_{\text{change}}, \tag{5}$$

$$\sum_{a \in \delta^{\text{out}}(e)} q_{ap} = 1 \qquad \forall e = (p - Org) \in \mathcal{E}_{\text{org}}, \tag{6}$$

$$\sum_{a \in \delta^{\text{out}}(e)} q_{ap} = \sum_{a \in \delta^{\text{in}}(e)} q_{ap} \qquad \forall p \in \mathcal{P}, e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}, \tag{7}$$

$$1 = \sum_{a \in \delta^{\text{in}}(e)} q_{ap} \qquad \forall e = (p - Dest) \in \mathcal{E}_{\text{dest}}, \tag{8}$$

$$t_p \geq x_e - M_2(1 - q_{ap}) \qquad \forall e = (p - Dest) \in \mathcal{E}_{\text{dest}}, a \in \delta^{\text{in}}(e), \tag{9}$$

$$z_a \in \{0, 1\} \qquad \forall a \in \mathcal{A}_{\text{change}}, \tag{10}$$

$$q_{ap} \in \{0, 1\} \qquad \forall p \in \mathcal{P}, a \in \mathcal{A}, \tag{11}$$

$$x_e \in \mathbb{N} \qquad \forall e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}, \tag{12}$$

$$t_p \in \mathbb{N} \qquad \forall p \in \mathcal{P}. \tag{13}$$

The objective function (1) minimizes the total delay of all passengers. Constraints (2) imply that events cannot take place earlier than in the original timetable and that source delays are taken into account. To make sure that delays are propagated through the network correctly, constraints (3) transfer the delay from the start of activity a to its end. For maintained connections, that is connections for which $z_a = 1$, constraints (4) transfer delays from the feeder train to the connecting train. The value of M_1 should be chosen large enough for these constraints to be correct. In Schöbel (2006) it has been shown that $M_1 = \max_{e \in \mathcal{E}} d_e$ is large enough. Constraints (2 - 4) are also present in the classical model.

Constraints (5 - 9) take the routing decisions into account. First of all, constraints (5) make sure that changing activities can only be used if the connection is maintained. Equations (6 - 8) are the constraints of the shortest path problem for each origin-destination pair p. For every pair, a path is selected from the origin $(p - Org) \in \mathcal{E}_{\text{org}}$ to the destination $(p - Dest) \in \mathcal{E}_{\text{dest}}$. The last constraint defines the arrival time for OD-pair p, where M_2 is again a large number. For the arrival event e that is selected and the driving activity a into this event, $q_{ap} = 1$, forcing that $t_p \geq x_e$ for this particular arrival event. All other path variables q_{ap} are equal to zero, therefore putting no restriction on the value of t_p.

To find the minimal value of M_2 for which (9) is correct, consider an arbitrary OD-pair $p \in \mathcal{P}$. As mentioned in Section 2, if $\max_{e' \in \mathcal{E}} d_{e'} \leq T$ only arrival events $e \in \mathcal{E}_{\text{arr}}$ for which $\pi_e \leq SP_p + n_p T + \max_{e' \in \mathcal{E}} d_{e'}$ should be connected to the destination event $(p - Dest)$, where

n_p is the number of transfers for these passengers if the timetable is operated as planned. Consider now an arbitrary arc $a = (e, (p - Dest)) \in \delta^{\text{in}}(p - Dest)$. For $\max_{e' \in \mathcal{E}} d_{e'} \leq T$ it holds that

$$x_e \leq \pi_e + \max_{e' \in \mathcal{E}} d_{e'} \leq SP_p + n_p T + 2 \max_{e' \in \mathcal{E}} d_{e'}.$$

Assuming that no passenger has more than two transfers, it follows that $M_2 = 2T + 2 \max_{e' \in \mathcal{E}} d_{e'}$ is large enough. Indeed, as

$$x_e - M_2 \leq SP_p + n_p T + 2 \max_{e' \in \mathcal{E}} d_{e'} - M_2 \leq SP_p + 2T + 2 \max_{e' \in \mathcal{E}} d_{e'} - M_2 = SP_p,$$

the constraint $t_p \geq x_e - M_2(1 - q_{ap})$ does not pose a restriction on t_p when $q_{ap} = 0$.
We remark that the variables z_a are not needed in the above model, since constraints (4) and (5) can be replaced by the constraints

$$x_e \geq x_{e'} + L_a - M(1 - q_{ap}) \quad \forall a = (e', e) \in \mathcal{A}_{\text{change}} \forall p \in \mathcal{P},$$

leading to an equivalent model. Nevertheless, we have chosen to leave these variables in the model to show the similarity with earlier models. Furthermore, the variables z_a could be used to guide the solution process.
In Section 5 we will use this formulation to analyze differences between delay management with re-routing of passengers and the classical delay management version without re-routing.

4 Special Cases of Delay Management with Re-Routing and their Complexities

In the previous section we gave an integer programming formulation for the general problem (DMwRR). Now we will identify simplifications and special cases of (DMwRR) in order to understand the border between still polynomial solvable and already NP-hard variants. The knowledge about the reasons for the NP-hardness as well as polynomial approaches for special cases can later serve to construct good heuristics for the general case.
In this section we will hence examine three special cases of (DMwRR). We first present a polynomial algorithm for the case of delay management with re-routing where the demand is given by only one OD-pair. We will then (slightly) generalize this case and allow that all OD-pairs start at the same origin but have different destinations. It will turn out even in this case delay management with re-routing is NP-hard. Finally, we will consider another variant with simplified delay costs. Although this is a strong simplification of delay management with re-routing, it will turn out to be NP-hard as well.

4.1 Delay management with re-routing for one single OD-pair

This subsection deals with a simplification of delay management with re-routing (DMwRR): We assume that we are given only one OD-pair $p = \{u, v, s_{uv}\}$. To simplify the notation in the following chapter we will identify $(p - Org)$ with u and $(p - Dest)$ with v, so u and v will be regarded as events in the network. In this case the problem is solvable by a modified version of Dijkstra's algorithm for finding a shortest path.

Let $\mathcal{N}$ be a network with feasible timetable π, $p = \{u, v, s_{uv}\}$ an OD-pair and $\mathcal{D}$ a set of source delays. Like in the original Dijkstra's algorithm we solve in every step the problem of determining an optimal path for a pair of events $\{u, i\}$ where $u = (p - Org)$ is the origin node of the OD-pair $p = \{u, v, s_{uv}\}$ under consideration and $i \in \mathcal{E}$. In order to do this formally, we need the following slight extension of (DMwRR):

Having in mind the practical application in passenger re-routing we defined in Section 2 the problem (DMwRR) for a network $\mathcal{N}$ and a set of OD-pairs $\mathcal{P}$ consisting of elements of the form $p = \{u, v, s_{uv}\}$ where u is the origin, v the destination and s_{uv} is the starting time. Now we also want to deal with OD-pairs as elements of the type $p^* = \{u, i, s_{uv}\}$ where $i \in \mathcal{E}$ is an *arbitrary* successor of u in $\mathcal{N}$. From a mathematical point of view we can do this easily by defining $t_{\mathcal{A}_{\text{fix}}}(p^*) := x_i$ as the (artificial) arrival time of such an OD-pair p^* in the network $\mathcal{N}(\mathcal{A}_{\text{fix}})$. We hence extend the problem (DMwRR) to instances consisting of a network $\mathcal{N}$ and a set of OD-pairs $\mathcal{P}$ of type p^*.

Let u be the origin node of the considered OD-pair. Determining an optimal path for a fixed pair of events $\{u, i\}$ can hence be seen as solving (DMwRR) for $\mathcal{N}$ and $\mathcal{P} = \{\{u, i, s_{uv}\}\}$.

The part of Dijkstra's algorithm that has to be modified is the calculation of the node labels that represent the earliest possible arrival times at the nodes. In order to calculate the transfer of delays efficiently, we define $tr[e]$ as the train belonging to an event $e \in \mathcal{E}_{\text{dep}} \cup \mathcal{E}_{\text{arr}}$.

According to Schöbel (2007), given a set of maintained connections $\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}$ the minimal arrival times considering the network $\mathcal{N}(\mathcal{A}_{\text{fix}})$ can be calculated iteratively as

$$x^{\mathcal{A}_{\text{fix}}}[e] = \max\{\pi[e] + d_e, \max_{i:(i,e)\in\mathcal{A}_{\text{drive}}\cup\mathcal{A}_{\text{wait}}\cup\mathcal{A}_{\text{fix}}} \{x^{\mathcal{A}_{\text{fix}}}[i] + L_{(i,e)}\}\}$$

using the critical path method.

Let $\tilde{\pi}[i] = x^{\emptyset}[i]$ for all $i \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$ denote the minimal arrival times calculated by the critical path method for the empty set of maintained connections. Set $\tilde{\pi}[u] = \tilde{\pi}[v] = s_{uv}$. We observe that for every set $\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}$ and every node $e \in \mathcal{E}_{\text{arr}} \cup \mathcal{E}_{\text{dep}}$

$$x^{\mathcal{A}_{\text{fix}}}[e] \geq \tilde{\pi}[e] \geq \pi[e] + d_e.$$

So we can equivalently determine the minimal arrival times for a given set $\mathcal{A}_{\text{fix}}$ as

$$x^{\mathcal{A}_{\text{fix}}}[e] = \max\{\tilde{\pi}[e], \max_{i:(i,e)\in\mathcal{A}_{\text{drive}}\cup\mathcal{A}_{\text{wait}}\cup\mathcal{A}_{\text{fix}}} \{x^{\mathcal{A}_{\text{fix}}}[i] + L_{(i,e)}\}\}.$$

In Lemma 1 and Lemma 2 we will prove properties of the optimal set of connections $\mathcal{A}_{\text{fix}}$ and the path used by the passengers in $\mathcal{N}(\mathcal{A}_{\text{fix}})$ for the case of one single OD-pair which will lead to a simplification of the calculation of the minimal arrival times for some $\mathcal{A}_{\text{fix}}$, that will be given in Lemma 3.
Lemma 1 states that in any optimal solution for $p^* = \{u, e, s_{uv}\}$ with $e \in \mathcal{E}$ only the connections on the path used from u to e have to be maintained.

Lemma 1 *Let $\tilde{\mathcal{A}}_{fix}$ be a set of maintained connections such that for a node $e \in \mathcal{E}$ the arrival time for $p^* = \{u, e, s_{uv}\}$ is minimal. Let $P = (\mathcal{E}_P, \mathcal{A}_P)$ be a path from u to e in $\mathcal{N}(\tilde{\mathcal{A}}_{fix})$. Then there exists an optimal set of maintained connections $\mathcal{A}^P_{fix}$ such that $\mathcal{A}^P_{fix} = \mathcal{A}_P \cap \mathcal{A}_{change}$.*

Proof Obviously $\tilde{\mathcal{A}}_{\text{fix}} \supset \mathcal{A}_P \cap \mathcal{A}_{\text{change}}$. On the other hand, if $a \in \tilde{\mathcal{A}}_{\text{fix}} \setminus \mathcal{A}_P$, no passenger on path P uses a, so we can remove it. □

The statement of Lemma 2 is the following: There is always a path with minimal arrival time such that the passengers use every train at most once.

Lemma 2 *Let e be a node in $\mathcal{N}$. For every set $\mathcal{A}_{fix}$ for which there exists a path from the origin u to e in $\mathcal{N}(\mathcal{A}_{fix})$, there also exists a path $P = (\mathcal{E}_P, \mathcal{A}_P)$ from u to e that fulfills the following condition:*
If $j, k \in \mathcal{E}_P$ such that $x^{\mathcal{A}_{fix}}[j] < x^{\mathcal{A}_{fix}}[k]$ and $tr[j] \neq tr[k]$, then $tr[j] \neq tr[l]$ for all $l \in \mathcal{E}_P$ with $x^{\mathcal{A}_{fix}}[k] < x^{\mathcal{A}_{fix}}[l]$.

Proof Assume that $\tilde{P}$ is a path from u to e in $\mathcal{N}(\mathcal{A}_{\text{fix}})$ such that $tr[l] = tr[j] \neq tr[k]$ and $x^{\mathcal{A}_{\text{fix}}}[j] < x^{\mathcal{A}_{\text{fix}}}[k] < x^{\mathcal{A}_{\text{fix}}}[l]$ for $j, l, k \in \tilde{P}$. Then the path from u to e that is identical to $\tilde{P}$ on all predecessors of j and successors of l but contains only nodes in $tr[j] = tr[l]$ between j and l is also contained in $\mathcal{N}(\mathcal{A}_{\text{fix}})$. Repeating this, we can exchange $\tilde{P}$ piecewise until we obtain a path P with the claimed property. □

The following lemma states that for computing the arrival time of a node of path P only the nodes of P are relevant.

Lemma 3 *For a node e, a path P from u to e in $\mathcal{N}(\mathcal{A}_{change})$ fulfilling the condition of Lemma 2 and the set $\mathcal{A}^P_{fix}[e] = \mathcal{A}_{change} \cap \mathcal{A}_P$ the minimal arrival time of node e can be calculated as*

$$x^{\mathcal{A}^P_{fix}[e]}[e] = \max\{\tilde{\pi}[e], x^{\mathcal{A}^P_{fix}[e]}[j] + L_{(j,e)}\}$$

for the predecessor j of e on the path P.

Proof If $(j,e) \in \mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}$, or if $(j,e) \in \mathcal{A}^P_{\text{fix}}[e]$ and (j,e) is the first trip of the train $tr[e]$, then (j,e) is the only arc terminating in e in $\mathcal{N}(\mathcal{A}^P_{\text{fix}}[e])$, due to $\mathcal{A}^P_{\text{fix}}[e] = \mathcal{A}_{\text{change}} \cap \mathcal{A}_P$. Thus

$$\begin{aligned} x^{\mathcal{A}^P_{\text{fix}}[e]}[e] &= \max\{\tilde{\pi}[e], \max_{i:(i,e)\in\mathcal{A}_{\text{drive}}\cup\mathcal{A}_{\text{wait}}\cup\mathcal{A}^P_{\text{fix}}[e]} \{x^{\mathcal{A}^P_{\text{fix}}[e]}[i] + L_{(i,e)}\}\} \\ &= \max\{\tilde{\pi}[e], x^{\mathcal{A}^P_{\text{fix}}[e]}[j] + L_{(j,e)}\}. \end{aligned}$$

Now let $(j,e) \in \mathcal{A}^P_{\text{fix}}[e]$ and $(k,e) \in \mathcal{A}$ with $tr[k] = tr[e]$. Due to Lemma 2 $x^{\mathcal{A}^P_{\text{fix}}[e]}[k] = \tilde{\pi}[k]$ and $\tilde{\pi}[e] \geq \tilde{\pi}[k] + L_{(k,e)} = x^{\mathcal{A}^P_{\text{fix}}[e]}[k] + L_{(k,e)}$. Using $\mathcal{A}^P_{\text{fix}}[e] = \mathcal{A}_{\text{change}} \cap \mathcal{A}_P$ it follows that

$$\begin{aligned} x^{\mathcal{A}^P_{\text{fix}}[e]}[e] &= \max\{\tilde{\pi}[e], \max_{i:(i,e)\in\mathcal{A}_{\text{drive}}\cup\mathcal{A}_{\text{wait}}\cup\mathcal{A}^P_{\text{fix}}[e]} \{x^{\mathcal{A}^P_{\text{fix}}[e]}[i] + L_{(i,e)}\}\} \\ &= \max\{\tilde{\pi}[e], x^{\mathcal{A}^P_{\text{fix}}[e]}[j] + L_{(j,e)}, x^{\mathcal{A}^P_{\text{fix}}[e]}[k] + L_{(k,e)}\} \\ &= \max\{\tilde{\pi}[e], x^{\mathcal{A}^P_{\text{fix}}[e]}[j] + L_{(j,e)}\}. \end{aligned}$$

□

Let's come back to our modified Dijkstra's algorithm. We solve problem (DMwRR) for different nodes i. In any iteration we store

- $T[i]$: Minimal arrival time in event i for passengers traveling from u to i with starting time s_{uv}.
- $\mathcal{A}_{\text{fix}}[i]$: Changing activities that have to be maintained in the optimal solution of (DMwRR) with OD-pair $\{u, i, s_{uv}\}$.
- $TD[i]$: Set of "forbidden trains" = trains that were used on the minimal path from u to i, not including $tr[i]$.

Let $PERM$ be the set of events for which (DMwRR) has been solved and the above values have been determined. For every e with a direct predecessor $i \in PERM$ we determine the preliminary arrival time

$$\tilde{T}[e] = \min_{i\in PERM:(i,e)\in\mathcal{A}, tr[e]\notin TD[i]} \{\tilde{\pi}[e], T[i] + L_{(i,e)}\}.$$

Like in Dijkstra's algorithm we fix the event $\hat{e}$ with smallest $\tilde{T}[e]$.
In order to calculate $\mathcal{A}_{\text{fix}}[\hat{e}]$ and $TD[\hat{e}]$ we distinguish two cases. Let $i_{\hat{e}}$ be the predecessor of $\hat{e}$ in the solution of (DMwRR) for $\{u, \hat{e}, s_{uv}\}$.

- If $\hat{a} = (i_{\hat{e}}, \hat{e})$ is a changing activity we obtain $\mathcal{A}_{\text{fix}}[\hat{e}] = \mathcal{A}_{\text{fix}}[i_{\hat{e}}] \cup \{(i_{\hat{e}}, \hat{e})\}$ and $TD[\hat{e}] = TD[i_{\hat{e}}] \cup \{tr[i_{\hat{e}}]\}$.

- Otherwise we simply set $\mathcal{A}_{\text{fix}}[\hat{e}] = \mathcal{A}_{\text{fix}}[i_{\hat{e}}]$ and $TD[\hat{e}] = TD[i_{\hat{e}}]$.

The algorithm is summarized below.

Algorithm: Modified Dijkstra for delay management with re-routing for one OD-pair

Input: Instance of (DMwRR) with network $\mathcal{N}$, feasible timetable π, delays d_e and one OD-pair $p = \{u, v, s_{uv}\}$.

Step 1. Generate the timetable $\tilde{\pi}$ by the critical path method. Set $\tilde{\pi}[u] = \tilde{\pi}[v] = s_{uv}$.

Step 2. Set $PERM = \{u\},\ TEMP = E \setminus \{u\},\ T[u] = s_{uv},\ \tilde{T}[e] = \infty$ for every $e \in TEMP$, $TD[u] = \emptyset,\ \mathcal{A}_{\text{fix}}[u] = \emptyset,\ \hat{e}^{\text{old}} = u$.

Step 3. For every $e \in TEMP$ such that $(\hat{e}^{\text{old}}, e) \in \mathcal{A}$, $tr[e] \notin TD[e^{\text{old}}]$
set $\tilde{T}[e] = \min\{\tilde{T}[e], \max\{\tilde{\pi}[e], T[\hat{e}^{\text{old}}] + L_{(\hat{e}^{\text{old}}, e)}\}\}$.

Step 4. Choose $\hat{e} \in \operatorname{argmin} \tilde{T}[e]$. Set $i_{\hat{e}}$ the corresponding predecessor of $\hat{e}$, $PERM = PERM \cup \{\hat{e}\},\ TEMP = TEMP \setminus \{\hat{e}\},\ T[\hat{e}] = \tilde{T}[\hat{e}]$.

Step 5. If $\hat{e} = v$ go to step 7.

Step 6. If $(i_{\hat{e}}, \hat{e}) \in \mathcal{A}_{\text{change}}$ set $\mathcal{A}_{\text{fix}}[\hat{e}] = \mathcal{A}_{\text{fix}}[i_{\hat{e}}] \cup \{(i_{\hat{e}}, \hat{e})\}$ and $TD[\hat{e}] = \{TD[i_{\hat{e}}] \cup \{tr[i_{\hat{e}}]\}\}$.
Otherwise set $\mathcal{A}_{\text{fix}}[\hat{e}] = \mathcal{A}_{\text{fix}}[i_{\hat{e}}]$ and $TD[\hat{e}] = TD[i_{\hat{e}}]$.
Set $\hat{e}^{\text{old}} = \hat{e}$. Go to step 3.

Step 7. Set $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{fix}}[v]$ and $t_v = T[v]$.

Output: Optimal set $\mathcal{A}_{\text{fix}}$ for the given instance of (DMwRR).

Theorem 1 *The algorithm finds an optimal solution $\mathcal{A}_{fix}$ to (DMwRR) with one OD-pair in time $O(n^2)$ where n is the number of nodes in the network $\mathcal{N}$.*

Proof First we observe that adding changing activities to a set A_1 does not influence the time for events e that happen before the added activities take place.
That means for two sets $A_1 \subset A_2 \subset \mathcal{A}_{\text{change}}$ and an event e the following statement holds:
If

$$x^{A_2}(e_1) \geq x^{A_2}(e) \text{ for all } (e_1, e_2) \in A_2 \setminus A_1,$$

it follows that

$$x^{A_1}[e] = x^{A_2}[e]. \tag{14}$$

Now we will show inductively that in every iteration of the algorithm for every node $e \in PERM$ it holds that

$$T[e] = x^{\mathcal{A}_{\text{fix}}[e]}[e]$$

for the labels $T[e]$ and the sets of changing activities $\mathcal{A}_{\text{fix}}[e]$ calculated by the algorithm.
As $T[u] = s_{uv} = x^{\emptyset}[u] = x^{\mathcal{A}_{\text{fix}}[u]}[u]$ the assumption holds for the origin u.
Assume that in the k-th iteration of the algorithm the assumption holds for the nodes in $PERM$. Let $\hat{e}$ be the node that is chosen in step 4 of the algorithm and $i_{\hat{e}} \in PERM$ such that $(i_{\hat{e}}, \hat{e}) \in \mathcal{A}$ and $\max\{\tilde{\pi}[\hat{e}], x^{\mathcal{A}_{\text{fix}}[i_{\hat{e}}]}[i_{\hat{e}}] + L_{(i_{\hat{e}},\hat{e})}\}$ is minimal. Let $\tilde{T}^{\text{old}}[\hat{e}]$ be the label of $\hat{e}$ at the beginning of step 3 and let $\hat{e}^{\text{old}}$ be the node that was added to $PERM$ in the $(k-1)$-th iteration. Then the new label of $\hat{e}$ is calculated as

$$\begin{aligned}\tilde{T}[\hat{e}] &= \begin{cases} \tilde{T}[\hat{e}^{\text{old}}] & \text{if } tr[\hat{e}] \in TD[\hat{e}^{\text{old}}] \\ \min\{\tilde{T}^{\text{old}}[\hat{e}], \max\{\tilde{\pi}[\hat{e}], T[\hat{e}^{\text{old}}] + L_{(\hat{e}^{\text{old}},\hat{e})}\}\} & \text{if } tr[\hat{e}] \notin TD[\hat{e}^{\text{old}}] \end{cases} \\ &= \min_{i \in PERM:(i,\hat{e}) \in \mathcal{A}, tr[e] \notin TD[i]} \max\{\tilde{\pi}[\hat{e}], T[i] + L_{(i,\hat{e})}\}.\end{aligned}$$

Let $\mathcal{A}^P_{\text{fix}} = \mathcal{A}_P \cap \mathcal{A}_{\text{change}}$ as in Lemma 3. We note that due to the construction of $\mathcal{A}_{\text{fix}}[i]$ for a node i

$$\mathcal{A}_{\text{fix}}[i] = \mathcal{A}^P_{\text{fix}}[i] \tag{15}$$

for the path P used by the Dijkstra algorithm as an optimal path from u to i.
Then

$$\begin{aligned}
T[\hat{e}] &= \min_{i \in PERM:(i,\hat{e}) \in \mathcal{A}, tr[e] \notin TD[i]} \max\{\tilde{\pi}[\hat{e}], T[i] + L_{(i,\hat{e})}\} && (16)\\
&= \min_{i \in PERM:(i,\hat{e}) \in \mathcal{A}, tr[e] \notin TD[i]} \max\{\tilde{\pi}[\hat{e}], x^{\mathcal{A}_{\text{fix}}[i]}[i] + L_{(i,\hat{e})}\} && (17)\\
&= \max\{\tilde{\pi}[\hat{e}], x^{\mathcal{A}_{\text{fix}}[i_{\hat{e}}]}[i_{\hat{e}}] + L_{(i_{\hat{e}},\hat{e})}\} && (18)\\
&= \max\{\tilde{\pi}[\hat{e}], x^{\mathcal{A}^P_{\text{fix}}[i_{\hat{e}}]}[i_{\hat{e}}] + L_{(i_{\hat{e}},\hat{e})}\} && (19)\\
&= \max\{\tilde{\pi}[\hat{e}], x^{\mathcal{A}^P_{\text{fix}}[\hat{e}]}[i_{\hat{e}}] + L_{(i_{\hat{e}},\hat{e})}\} && (20)\\
&= x^{\mathcal{A}^P_{\text{fix}}[\hat{e}]}[\hat{e}] && (21)\\
&= x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}] && (22)
\end{aligned}$$

where we use (15) in (19) and (22). (17) holds because of the assumption $T[i] = x^{\mathcal{A}_{\text{fix}}[i]}[i]$ for all $i \in PERM$ and (20) is true due to the initial observation (14).
It remains to show that the set $\mathcal{A}_{\text{fix}}[\hat{e}]$ and the label $T[\hat{e}] = x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}]$ are optimal for the node $\hat{e}$ chosen in step 4 of the algorithm, that means that there is no $A \subset \mathcal{A}_{\text{change}}$ such that there

is a path from u to $\hat{e}$ in $\mathcal{N}(A)$ and

$$x^A[\hat{e}] < x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}].$$

This assumption will also be proven inductively. For the origin node u setting $\mathcal{A}_{\text{fix}}[u] = \emptyset$ leads to $T[u] = s_{uv}$ which is optimal.
Suppose that in the iterations 1 to $k-1$ of the algorithm the choice of $\mathcal{A}_{\text{fix}}[e]$ and the labels $T[e]$ are optimal for the regarded nodes e.
Now let $\hat{e}$ be the node chosen in step 4 in the k-th iteration, i.e. such that $T[\hat{e}] = \tilde{T}[\hat{e}] \leq \tilde{T}[e]$ for every $e \in TEMP$. Suppose that there is a set $A \subset \mathcal{A}_{\text{change}}$ such that there is a path from u to $\hat{e}$ in $\mathcal{N}(A)$ and

$$x^A[\hat{e}] < x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}]. \tag{23}$$

Let $P^A_{u\hat{e}}$ be an optimal path from u to $\hat{e}$ in $\mathcal{N}(A)$, that satisfies the conditions of Lemma 2.

1. If the predecessor e_0 of $\hat{e}$ in $P^A_{u\hat{e}}$ is in $PERM$, because of the assumption that the labels $T[e]$ and chosen sets $\mathcal{A}_{\text{fix}}[e]$ are optimal for all $e \in PERM$

$$\begin{aligned} x^A[\hat{e}] &= \max\{\tilde{\pi}[\hat{e}], T[e_0] + L_{(e_0,\hat{e})}\} \\ &\geq \min_{i \in PERM:(i,\hat{e})\in\mathcal{A}, tr[\hat{e}]\notin TD[i]} \max\{\tilde{\pi}[\hat{e}], T[i] + L_{(i,\hat{e})}\} \\ &= T[\hat{e}] = x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}]. \end{aligned}$$

   which contradicts (23).

2. If the predecessor e_0 of $\hat{e}$ in $P^A_{u\hat{e}}$ is in $TEMP$, let e_1 denote the last node in $PERM$ on the path $P^A_{u\hat{e}}$ (e_1 exists because $u \in PERM$) and $e_2 \in TEMP$ its successor. So as $T[\hat{e}] = \tilde{T}[\hat{e}] \leq \tilde{T}[e]$ for every $e \in TEMP$

$$\begin{aligned} x^A[\hat{e}] &> x^A[e_2] \\ &\geq \max\{\tilde{\pi}[e_2], T[e_1] + L_{(e_1,e_2)}\} \\ &\geq \min_{i \in PERM:(i,e_2)\in\mathcal{A}, tr[e_2]\notin TD[i]} \max\{\tilde{\pi}[e_2], T[i] + L_{(i,e_2)}\} \\ &= \tilde{T}[e_2] \geq \tilde{T}[\hat{e}] = T[\hat{e}] = x^{\mathcal{A}_{\text{fix}}[\hat{e}]}[\hat{e}] \end{aligned}$$

   which contradicts (23).

Now it remains to show that $\mathcal{A}_{\text{fix}} = \mathcal{A}_{\text{fix}}[v]$ and $t_{\mathcal{A}_{\text{fix}}}(p) = T[v]$ is an optimal solution to (DMwRR) for the OD-pair $p = \{u, v, s_{uv}\}$. As defined in Section 2, $\mathcal{A}_{\text{fix}}[v]$ is optimal if it minimizes $t_{\mathcal{A}_{\text{fix}}}(p) = x^{\mathcal{A}_{\text{fix}}}[e]$ for the predecessor e of v on a minimal path from u to v in the network $\mathcal{N}(\mathcal{A}_{\text{fix}})$. Suppose that the set $\mathcal{A}_{\text{fix}}[v]$ and the predecessor e calculated by the

algorithm are not optimal with regard to an optimal path from u to v. Let A be an optimal set, P_{uv}^A an optimal path in $\mathcal{N}(A)$ and e_0 the optimal predecessor. Then

$$t_A(p) < t_{\mathcal{A}_{\text{fix}}}(p). \tag{24}$$

1. If $e_0 \in PERM$, because of the assumption that the labels $T[i]$ and chosen sets $\mathcal{A}_{\text{fix}}[i]$ are optimal for all $i \in PERM$

$$\begin{aligned} t^A[v] &= \max\{\tilde{\pi}[v], x^A[e_0] + L_{(e_0,v)}\} \\ &\geq \min_{i \in PERM:(i,v)\in\mathcal{A}} \max\{\tilde{\pi}[v], T[i] + L_{(i,v)}\} \\ &= \min_{i \in PERM:(i,v)\in\mathcal{A}} \max\{s_{uv}, T[i] + L_{(i,v)}\} \\ &= T[v] \\ &= t^{\mathcal{A}_{\text{fix}}[v]}[v]. \end{aligned}$$

   which contradicts (24).

2. If the predecessor e_0 of v in P_{uv}^A is in $TEMP$ let e_1 denote the last node in $PERM$ on the path P_{uv}^A and $e_2 \in TEMP$ its successor. So as $T[v] = \tilde{T}[v] \leq \tilde{T}[e]$ for $e \in TEMP$

   $$t^A[v] = x^A[e_0] > x^A[e_2] = \max\{\tilde{\pi}[e_2], T[e_1] + L_{(e_1,e_2)}\} \geq \tilde{T}[e_2] \geq \tilde{T}[v] = T[v] = x^{\mathcal{A}_{\text{fix}}[v]}[v]$$

   which contradicts (24).

The generation of the timetable in step 1 is done in time $O(n^2)$ by the procedure given in Schöbel (2007). Inspecting step 3, we note that for the setting of the labels $\tilde{T}$, summing up over all iterations every arc $a \in \mathcal{A}$ has to be considered at most once. As the steps $4-6$ are done in time $O(n)$, steps $3-6$ can be executed in $O(n^2)$. As step 7 is also in $O(n^2)$, the running time of the modified Dijkstra algorithm is $O(n^2)$.

□

We can use this algorithm to determine a lower bound on the optimal solution for the general (DMwRR) problem as follows: We apply the modified Dijkstra algorithm for delay management with re-routing with one OD-pair for every OD-pair $p \in \mathcal{P}$ and sum up over the solution values weighted with w_p. This gives us a lower bound on the solution of (DMwRR).

Lemma 4 *Let (DMwRR) be given together with a set of OD-pairs $\mathcal{P}$ and let z^* be its optimal objective value.*
For any OD-pair $p \in \mathcal{P}$ let f_p denote the objective value of the reduced problem (DMwRR)

with only the OD-pair $p = \{u, v, s_{uv}\}$. *Then*

$$t_p \geq f_p$$

is a valid inequality for our integer programming formulation for (DMwRR) for any $p \in \mathcal{P}$. *In particular, we have*

$$\sum_{p \in \mathcal{P}} w_p f_p \leq z^*.$$

Moreover, this bound on the objective value can be calculated in in $O(|\mathcal{P}|n^2)$.

This bound significantly improves the time needed to solve the integer programm for (DMwRR) as will be shown in Section 5.

4.2 Delay management with re-routing for a tree-like structure of the demand

In Section 4.1 we have seen that for the case of only one OD-pair, (DMwRR) is solvable in polynomial time and we have given a Dijkstra-type solution algorithm for this case. Dijkstra-type algorithms can usually be generalized to trees. We hence investigate the question if the case of multiple OD-pairs in which all passengers have the same origin and the same starting time can still be solved by the approach of the previous section. However, this generalization of our problem already turns out to be strongly NP-hard.

Theorem 2 *Delay management with re-routing is strongly NP-hard, even if only one delay occurs, all origin and destination nodes are connected to only one event in the network and all OD-pairs* $\{u_k, v_k, s_{u_k v_k}\}$ *have the same origin* $u_k := u$ *and the same starting time* $s_{u_k v_k} := 0$.

Proof This theorem will be proven by reduction to the NP-complete decision problem *minimum cover* (see Garey and Johnson (1979)). An instance of minimum cover consists of a finite set $S = \{s_j : j = 1, \ldots, n\}$, a collection $C = \{c^i : i = 1, \ldots, m\}$ of subsets of S and a positive integer $K \leq |C|$. The question to decide is whether there is a subset C' of C with $|C'| \leq K$ such that every element of S is contained in at least one element of C'. The structure of the minimum cover problem can be represented by a matrix $M = (m_{ij})_{i=1,\ldots,m, j=1,\ldots,n}$ with

$$m_{ij} = \begin{cases} 1 & \text{if } s_j \in c^i \\ 0 & \text{otherwise.} \end{cases}$$

We construct an instance of the delay management problem in which S corresponds to the OD-pairs and C to connections for which we have to decide whether they are maintained or not. We have to cover all OD-pairs (i.e. make sure that all passengers reach their destinations) with a minimal set of maintained connections since maintaining a connection causes

costs (represented as delays) to other passengers. Our construction is the following:

For a given instance (S, C, K) of minimum cover we transfer the matrix M to a set of stations $A = \{a_{ij} : m_{ij} = 1\}$, that means whenever $m_{ij} = 1$ for $1 \leq i \leq m$ and $1 \leq j \leq n$ there is a station a_{ij}. There are trains tr^i for $i = 1, \ldots, m$ starting all at a station a_0 and running through the existing stations a_{ij} in increasing order of j and trains tr_j running through the existing stations a_{ij} in increasing order of i. The stations a_{ij} offer the possibility to change from train tr^i to train tr_j, so if and only if $m_{ij} = 1$, it is possible to change from tr^i to tr_j. There are no slack times on the driving or waiting activities of the trains tr^i and tr_j, as well as on the changing activities between these trains, so any delay of a train tr^i will propagate to all following events of the train and to the events of the trains tr_j if the changing activity between these trains at station a_{ij} is maintained.
There are destinations v_j for $j = 1, \ldots, n$ that are reached by train tr_j after the last station a_{ij} that this train passes.
We introduce an origin u with one train starting there, going to a station a_0. At station a_0 there are changing activities to the departure events of the trains tr^i, having no slack times. In this construction, the passengers wanting to travel from u to v_j at station a_0 have to choose a train tr^i such that they will be able to change to the train tr_j and reach their destination, that means a train tr^i such that a_{ij} exists.
We suppose that a delay of 1 at the arrival event of train tr^0 at station a_0 occurs. As there are no slack times on the changing activities to the trains tr^i, if a connection to a train tr^i is maintained, this train will receive a delay of 1. When passengers change to a train tr_j at station a_{ij}, this delay will be transferred to the train tr_j, thus all passengers of the OD-pairs $\{u, v_j, 0\}$ arrive at their destinations with a delay of 1.
Now we assume that there are more OD-pairs $\{u, v^1, 0\}, \ldots, \{u, v^m, 0\}$ that can reach their destination via the trains tr^i. To this end, for every $i = 1, \ldots, m$ we introduce two more stations a_{in+1} and v^i. We assume that the train tr^0 after leaving a_0 runs through the stations a_{in+1} for all $i = 1, \ldots, m$. At the driving activity from a_0 to a_{1n+1} we set the slack time to be 1, thus when the train arrives at station a_{1n+1}, despite of the delay of 1 occurring at the arrival event in a_0, it is not delayed any more. We allow the passengers to change from tr^0 to tr^i at every station a_{in+1}. So we can assume that all passengers wanting to travel from u to v^i for an $i = 1, \ldots, m$ will take train tr^0 until station a_{in+1} and thus will be delayed if and only if the connection from tr^0 to tr^i at station a_0 is maintained. We will denote the constructed event activity network by $\mathcal{N}$.
In Figure 3 the station network for an instance of DMwRR constructed from an instance of minimal cover with $S = \{s_1, s_2\}$ and $C = \{\{s_1\}, \{s_2\}, \{s_1, s_2\}\}$ is pictured, while you can find the detailed event activity network $\mathcal{N}$ in Figure 4.

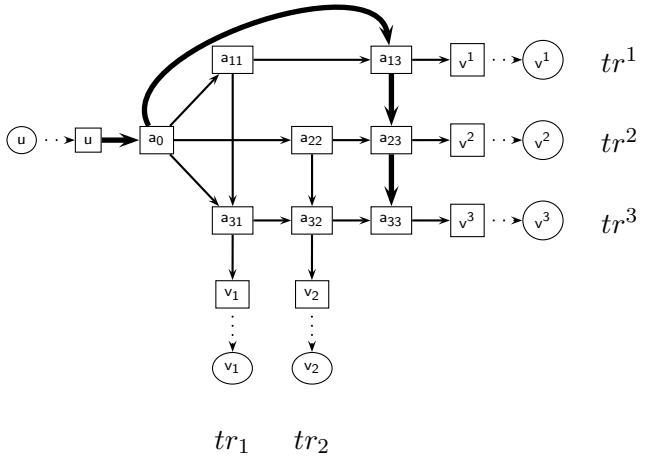

Figure 3: The station network for the instance of the delay management problem with re-routing constructed from an instance of minimal with $S = \{s_1, s_2\}$ and $C = \{\{s_1\}, \{s_2\}, \{s_1, s_2\}\}$. The square nodes are the departure and arrival events. The origin and destination events are represented by ovals. There are six trains, tr^0 represented by the thick line, tr^1, tr^2 and tr^3 starting at station a_0 and going from left to right and tr_1 and tr_2 going top down.

Setting the number of passengers to be $w_{uv^i} = w_{uv_j} = 1$, we can now show that the instance (S, C, K) of minimum cover has a solution if and only if there is a set $\mathcal{A}_{\text{fix}}$ such that the sum over the delays of the OD-pairs in the network $\mathcal{N}(\mathcal{A}_{\text{fix}})$ is smaller than or equal to $\tilde{K} = n + K$:
We divide $\mathcal{A}_{\text{change}}$ into two sets: the changing activities at station a_0 $\mathcal{A}^1_{\text{change}} := \{(tr^0 - a_0 - Arr, tr^i - a_0 - Dep) : i = 1, \ldots, m\}$ and all other changing activities $\mathcal{A}^2_{\text{change}} := \{(tr^i - a_{ij} - Arr, tr_j - a_{ij} - Dep)\} \cup \{(tr' - a_{in+1} - Arr, tr^i - a_{in+1} - Dep)\}$. Now we observe that maintaining a connection in $\mathcal{A}^2_{\text{change}}$ does not yield a delay for any OD-pair. So we choose to maintain all connections in $\mathcal{A}^2_{\text{change}}$.
For a solution C' of minimum cover we set $\mathcal{A}_{\text{fix}}(C') := \mathcal{A}^2_{\text{change}} \cup \{(tr^0 - a_0 - Arr, tr^i - a_0 - Dep) : c^i \in C'\}$ and vice versa for a solution $\mathcal{A}_{\text{fix}} \supset \mathcal{A}^2_{\text{change}}$ we define $C'(\mathcal{A}_{\text{fix}}) = \{c^i : (tr^0 - a_0 - Arr, tr^i - a_0 - Dep) \in \mathcal{A}_{\text{fix}}\}$. Thus we have a bijection between subsets $C' \subset C$ and $\mathcal{A}_{\text{fix}} \supset \mathcal{A}^2_{\text{change}}$.
Let $A \subset \mathcal{A}^1_{\text{change}}$. We see that in $\mathcal{N}(A \cup \mathcal{A}^2_{\text{change}})$ there exists a path from u to v_j if and only if at least for one i with $s_j \in c^i$ the connection $(tr^0 - a_0 - Arr, tr^i - a_0 - Dep)$ is maintained. Thus a set $\mathcal{A}_{\text{fix}}$ is feasible for DMwRR (that means for every OD-pair there exists a path from origin to destination) if and only if the corresponding set C' is feasible for minimum cover.
Furthermore we observe that the OD-pairs $\{\{u, v_j, 0\} : j = 1, \ldots, n\}$ will reach their destination with a delay of 1, because there are no slack times on the paths from u to v_j for all $j = 1, \ldots, n$. For the other OD-pairs $\{u, v^i, 0\}$ the delay is 1 if the connection

$(tr^0 - a_0 - Arr, tr^i - a_{ij} - Dep)$ is maintained and 0 otherwise.
Thus for a pair of solutions $\mathcal{A}_{\text{fix}} \supset \mathcal{A}^2_{\text{change}}$ and $C' \subset C$ with solution values $z(\mathcal{A}_{\text{fix}})$ and $z(C')$

$$z(\mathcal{A}_{\text{fix}}) = z(C') + n.$$

That means there is a solution to the constructed instance of DMwRR with solution value $\leq \tilde{K}$ if and only if there is a solution to (S, C, K) with solution value $\leq K$.
So every instance of minimum cover can be transformed to an instance of DMwRR with the claimed properties. □

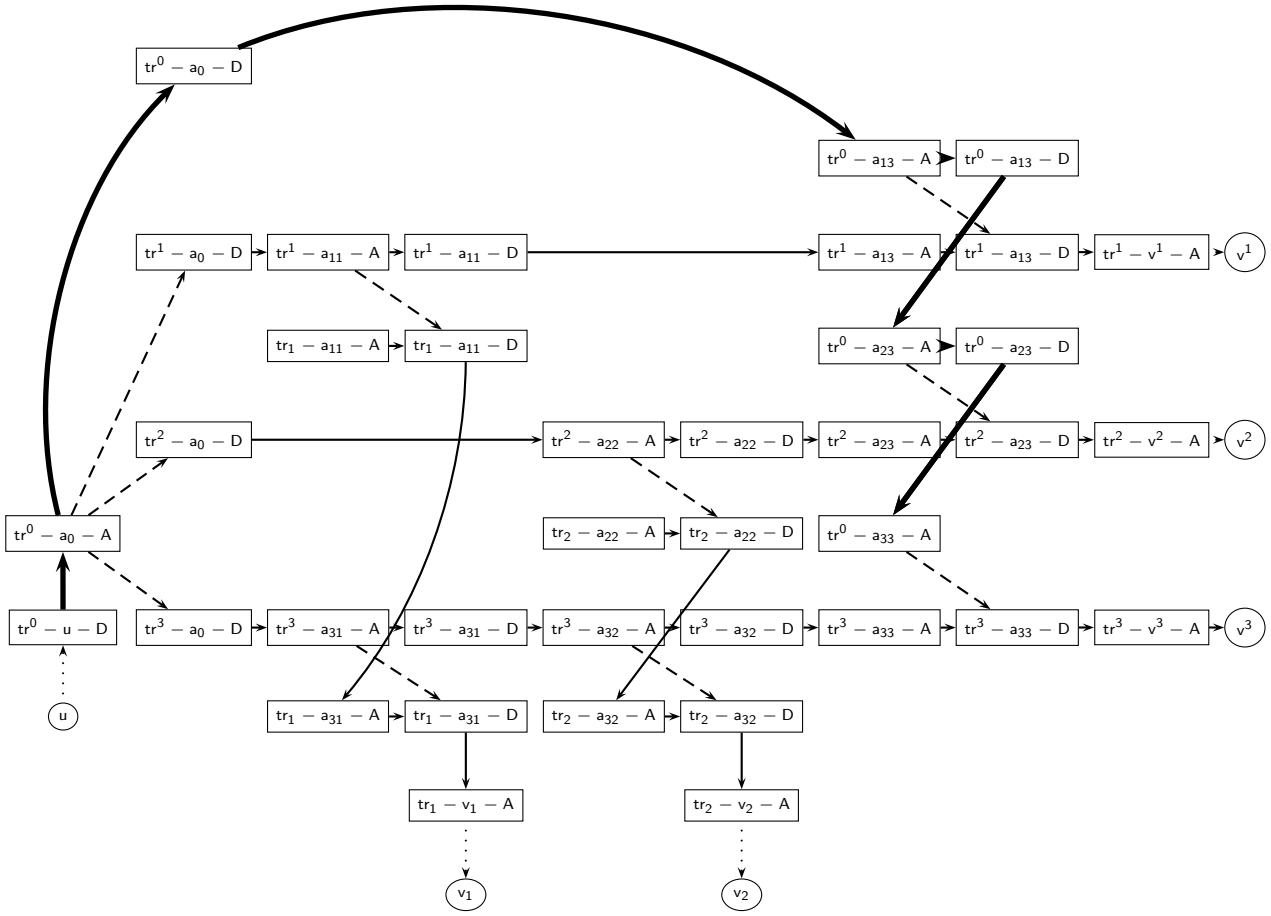

Figure 4: The event activity network for the instance of the delay management problem with re-routing constructed from an instance of minimum cover with $S = \{s_1, s_2\}$ and $C = \{\{s_1\}, \{s_2\}, \{s_1, s_2\}\}$. The square nodes are the departure and arrival events. The origin and destination events are represented by ovals. The dotted lines are the origin and destination arcs, the solid lines represent driving and waiting activities, changing activities are represented by dashed lines. There are six trains, tr^0 represented by the thick line, tr^1, tr^2 and tr^3 starting at station a_0 and going from left to right and tr_1 and tr_2 going top down.

4.3 Re-routing with simplified costs

The delays that arise in delay management with re-routing for the passengers by the wait-depart decisions for the connections can be divided into two types:

1. A connection is maintained: The waiting train and the passengers on the waiting train are delayed.

2. A connection is not maintained: The passengers that wanted to take this connection have to travel along another, probably longer path.

Calculating the delay of the first type by a heuristic approach motivates the following re-routing problem with simplified costs:
Let $\mathcal{N} = \{\mathcal{E}, \mathcal{A}\}$ be a directed network with edge lengths L_a for all $a \in \mathcal{A}$. Let $\mathcal{A}_{\text{change}} \subset \mathcal{A}$ be a set of connections that can be maintained or removed. We assume that maintaining a connection $a \in \mathcal{A}_{\text{change}}$ yields a fixed delay of d_a for the passengers. This delay can be regarded as a cost for opening a connection that is added to the solution value. Let $\mathcal{P}$ be a set of OD-pairs, given as a subset of $\mathcal{E} \times \mathcal{E}$ with demand w_p for each $p = \{u, v\} \in \mathcal{P}$. The objective of this variant is to minimize the simplified costs arising as fixed delays for maintaining connections plus the travel costs of the OD-pairs. Hence, the objective function is

$$\min_{\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}} \sum_{p \in \mathcal{P}} w_p \cdot D_{\mathcal{A}_{\text{fix}}}(u, v) + \sum_{a \in \mathcal{A}_{\text{fix}}} d_a$$

where $D_{\mathcal{A}_{\text{fix}}}(u, v) = \sum_{a \in P_{uv}} L_a$ with P_{uv} being a shortest path from u to v in the network in which all connections $a \in \mathcal{A}_{\text{change}} \setminus \mathcal{A}_{\text{fix}}$ are removed.
In contrast to (DMwRR), in this simplified variant we are trying to minimize shortest path distances regarding the edge lengths L_a and not arrival times. The simplified delays arise as costs or penalties, whenever a connection $a \in \mathcal{A}_{\text{fix}}$ is maintained and do not influence other parts of the network, while in (DMwRR) they can propagate through big parts of the network and delay the following events.
Like in delay management with re-routing this problem can be solved in polynomial time if there is only one OD-pair (by adding the simplified costs d_a divided by the demand of the OD-pair w_p for a connection a to its length L_a and applying Dijkstra's algorithm). However, by modifying the proof of Theorem 2 it can be shown that even this simplified variant is strongly NP-hard.

Theorem 3 *Re-routing with simplified costs is strongly NP-hard, even if all origin and destination nodes are connected to only one event in the network and all OD-pairs $\{u_k, v_k\}$ have the same origin $u_k := u$.*

Proof Analogously to the proof of strong NP-hardness for (DMwRR) we can prove this theorem by constructing an equivalent re-routing with simplified costs problem for each instance of minimum cover. For this proof we simplify the network $\mathcal{N}$ from the proof of Theorem 2 to a network $\tilde{\mathcal{N}}$ by removing all events at the stations a_{in+1} and v^i, the destination nodes v^i, and all related activities. The set of OD-pairs is $\mathcal{P} = \{\{u, v_j\} : j = 1, \ldots, m\}$ with unit demand. For the changing activities from tr^0 to tr^i we set the simplified costs to 1, for the ones from tr^i to tr^j to 0. Similar to the proof of Theorem 2 we observe that we can assume the connections $(tr^i - a_{ij} - Arr, tr_j - a_{ij} - Dep)$ to be maintained because their simplified costs are 0. Like in that proof for a given set of C' of subsets of S we define

$$\mathcal{A}_{\text{fix}}(C') := \{(tr^0 - a_0 - Arr, tr^i - a_0 - Dep) : c_i \in C'\} \cup \{(tr^i - a_{ij} - Arr, tr_j - a_{ij} - Dep) : m_{ij} = 1\}$$

and for a given subset $\mathcal{A}_{\text{fix}} \supset \{(tr^i - a_{ij} - Arr, a_{ij} - tr_j - Dep) : m_{ij} = 1\}$ we set

$$C'(\mathcal{A}_{\text{fix}}) = \{c_i : (tr^0 - a_0 - Arr, tr^i - a_0 - Dep) \in \mathcal{A}_{\text{fix}}\}.$$

Now a set C' of subsets of S and the associated subset $\mathcal{A}_{\text{fix}} \subset \mathcal{A}_{\text{change}}$ are both feasible or infeasible and have the same objective value as can be seen analogously to the proof of Theorem 2.

□

5 Computational Experiments

We have created four cases to evaluate the integer programming formulation given in Section 3. We will first present the cases that we considered. Then we will discuss the solution of the integer programs and show that the polynomial algorithm for one OD-pair can be applied to improve the overall running times. Finally, we will demonstrate that taking re-routing into account explicitly reduces the total delay significantly.

5.1 Cases

In all cases we consider a part of the railway network in the Netherlands during a period in the late evening. The first case corresponds to the example described in Section 1. The stations that are included in the first case are represented by black circles in Figure 5. We focus on the situation where the train from Zwolle arrives in Amersfoort with a delay. Therefore, only a short time period of 2 hours in the evening is taken into account. As described in Section 1, three intercities and one regional train are considered. The second case considers the stations that are indicated by a grey or black circle in Figure 5. All long distance trains are considered for a time period of 4 hours. The third case contains all long distance trains in the Randstad,

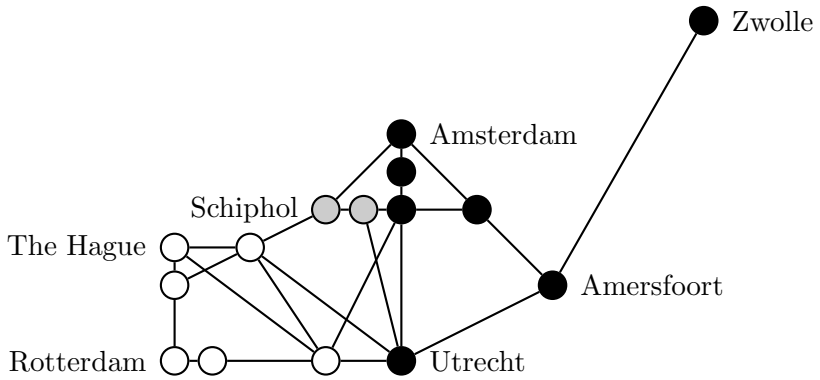

Figure 5: The railway network that is considered in the numerical experiments. The western part of the country is depicted. A circle in the picture indicates a station where long distance trains stop. The stations where only regional trains stop are not depicted. A line indicates that there is a direct connection between two stations. For each line, there are two or four intercities and two regional trains per hour.

which is the Western, most populated part of the Netherlands. This case includes all stations in Figure 5. The railway network in this area is very dense, as can be seen from the picture. Note that in the second and third case, the regional trains are not considered. Finally, the fourth case is an extension of the second one. It includes also the regional trains. When the regional trains are taken into account, the number of transfer arcs grows enormously, because much more trains depart at each station during a given time interval. In the fourth case, we have only included the OD-pairs with high passenger figures. Note that Figure 5 shows only a part of the railway network in the Netherlands.

The timetable and the passenger figures are obtained from Netherlands Railways. For the first case only a delay for the train from Zwolle to Amersfoort is interesting, because otherwise no connections are violated. This delay can take values between 0 and 30 minutes. For each other case we have generated 100 delay scenarios. In each scenario, each arrival event has a probability of 10% to be delayed. If a train is delayed, the size of this delay is a uniformly distributed integer number between 1 and 15 minutes.

In Table 1, we present some information about the cases and the sizes of the resulting event-activity networks. The second column gives the number of OD-pairs that are included. Recall that an OD-pair $p \in \mathcal{P}$ is characterized by its origin and destination station and the start time. If it is possible to travel from one station to another at several times, these possibilities correspond to multiple OD-pairs. To each OD-pair $p \in \mathcal{P}$ we associate a passenger figure w_p. The third column gives the total number of passengers that are considered in each case. The number of passengers is scaled for secrecy and does not represent the true number of passengers that travel in the given time period. For both the number of OD-pairs and the number of passengers, we have reported the total number and the percentage that need a

Case	OD-pairs	Passengers	Trains	Size of the event-activity network					Binary Variables
				$\lvert\mathcal{E}_{\text{dep}}\rvert$	$\lvert\mathcal{A}_{\text{op}}\rvert$	$\lvert\mathcal{A}_{\text{ch}}\rvert$	$\lvert\mathcal{A}_{\text{org}}\rvert$	$\lvert\mathcal{A}_{\text{dest}}\rvert$	
I	111 (15%)	23 (2.3%)	7	28	49	10	207	219	6585
II	283 (56%)	145 (15%)	117	184	251	870	3739	3102	324954
III	675 (48%)	341 (21%)	168	314	460	1493	9276	7201	1336245
IV	239 (4%)	190 (4%)	282	1022	1760	8068	7725	7586	2372271

Table 1: The cases investigated: For each case, the number of OD-pairs, the number of passengers and the number of trains are presented. Furthermore, the size of the event-activity network that models each case is given. Note that $|\mathcal{E}_{\text{arr}}| = |\mathcal{E}_{\text{dep}}|$ and that $\mathcal{A}_{\text{op}} = |\mathcal{A}_{\text{drive}} \cup \mathcal{A}_{\text{wait}}|$.

transfer. The fourth column indicates the number of trains. If a train runs from station A to B and continues from station B to station C, and so on, these trips are counted as one train. Finally, the last columns present the dimensions of the corresponding event-activity networks and the number of binary variables in the resulting integer program.

Comparing Cases II and IV, we see that the number of trains is doubled, while the number of transfer arcs in the event-activity network is ten times as large. We also observe that there are far more departure and arrival events, as the regional trains stop more often than the long distance trains. Regarding the passengers' data, we see that roughly half of the OD-pairs in Case II and Case III need a transfer, but that the percentage of passengers who transfer is only about 20 percent. The low number of trips and passengers with a transfer in Case IV is a consequence of only considering the OD-pairs with high passenger figures. If all OD-pairs were considered, the percentage of trips with a transfer would be 16 and the percentage of passenger who have to transfer would equal 12. This is in line with the global percentage: In the Netherlands about 20 percent of the passengers transfer during their trips.

5.2 Computational results

We used CPLEX 11.1 on an Intel Xeon Quad PC (3.0 GHz) with 3 GB of memory to solve the mathematical models presented in Section 3. Table 2 reports the characteristics of the optimal solutions for the first three cases. The fourth case could not be solved. Each entry in the table is the average value over all delay scenarios. The second column gives the average objective value. Then, the number of events with a delay and the number of dropped connections are reported. The next columns give the percentage of OD-pairs and the percentage of passengers that have a delay.

The first case can be solved within one second. To give some insight into the structure of the solutions, we will describe the routes for passengers that want to travel from Zwolle to Amsterdam. At Amersfoort, the intercity to Amsterdam waits at most one minute for the delayed train from Zwolle. If the train from Zwolle arrives later, passengers should travel

Case	Objective Value	Delayed events	Dropped connections	Delayed OD-pairs	Delayed passengers
I	30462	2.7	1.9	11.4 %	9.0 %
II	172073	29.7	13.1	21.9 %	16.0 %
III	523978	55.4	20.2	23.8 %	21.4%

Table 2: Delay management with re-routing: For each case, the objective value and the time to solve the integer program are given. Furthermore, the average number of delays, dropped connections, and delayed trips and passengers are given.

	$t_p \geq SP_p$		$t_p \geq f_p$	
Case	Average (s)	Max (s)	Average (s)	Max (s)
II	3.1	6.1	2.3	4.2
III	15.3	26.0	11.7	18.7

Table 3: The average and maximal running times for different lower bounds on the arrival times t_p. The running times include the time needed to obtain the bound.

via Utrecht and transfer there. However, if the connection in Utrecht cannot be made, the passengers should transfer in Amersfoort instead and use the regional train to Amsterdam. This reveals that the optimal routes for passengers are very sensitive to the delays of the trains.

The second and third case can be solved within one minute. The fourth case cannot be solved. Very early in the solution process, CPLEX runs out of memory and quits, even before the first feasible solution was found. The integer program resulting from the fourth case is too large to be solved with CPLEX. This indicates that more sophisticated models and solution approaches are needed to solve larger cases. Developing reduction techniques to reduce the sizes of the event-activity networks is one possible direction for future research.

The arrival times of the passengers for an OD-pair $p \in \mathcal{P}$ is bounded from below by SP_p, as passengers cannot arrive earlier than planned. As explained in Lemma 4, the polynomial algorithm for one OD-pair $p \in \mathcal{P}$ can be used to find a better lower bound on the arrival time t_p. For the second and third case, we have evaluated the effect on the computation time of adding the inequality

$$t_p \geq f_p$$

for every $p \in \mathcal{P}$ to the integer program. As can be seen in Table 3, the average running time for both cases is reduced by 25 percent. The maximum running times are reduced by 30 percent. We conclude that applying the polynomial algorithm to obtain better bounds on the arrival times improves the solution process.

	Objective Value			Dropped connections		
Case	NW	DM	DMwRR	NW	DM	DMwRR
I	31217	31817	30462	2.2	1.7	1.9
II	186988	172834	168799	19.2	13.9	13.0
III	559612	532705	515099	31.2	21.8	20.3

Table 4: Comparison of different models. Model NW implements a no-wait policy. Model DM is the classical delay management model. Model DMwRR is our model which includes re-routing.

5.3 The impact of passenger re-routing

For the first three cases, we have compared the performance of our model to a no-wait policy and to the classical model without re-routing from Schöbel (2007). In a no-wait policy, all trains depart as early as possible. The timetable is then determined by the operational constraints only. The no-wait policy is denoted by NW. The classical model assumes that a passenger that misses a connection will have a delay of one cycle time. We denote this model by DM. For both NW and DM, we have implemented the model only to decide which connections to maintain. Given these wait-depart decisions, the passengers are then re-routed to find their actual delay. In this way, a fair comparison can be made between the policies.
As we consider a time period in the late evening, some passengers miss their connection to the last train if re-routing is not included in the optimization of the wait-depart decisions. It is impossible for these passengers to arrive at their destination. Our model, that takes re-routing into account during the optimization of the wait-depart decisions, finds a route for all passengers explicitly. This ensures that all passengers can arrive at their destination, which is a clear advantage of taking re-routing into account. It is hard to assign a specific delay to the passengers for which no route is found. To be able to compare the delay in the three models, we therefore excluded these passengers from our data. To do so, we first evaluated the no-wait policy. Then we determined which passengers could not arrive at their destination and removed them from our input. In the first case, a route could be found for all passengers. In the second case, on average 0.17 passengers were excluded. In the third case 0.37 passengers were removed.
Table 4 reports the characteristics of the solutions for the cases without the excluded passengers. For each case, we report the average objective value and the number of missed connections.
For the first case, even the no-wait policy performs better than the classical model. Note that this case is specifically selected to explain why re-routing should be taken into account, so it could be expected that the classical model without explicit re-routing does not perform well on this case. The model with re-routing gives the best solutions. On average the delay is reduced by 4.4 percent with respect to the classical model and by 2.4 percent with respect

to a no-wait policy. The model with re-routing drops slightly more connections than the model without re-routing. This corresponds to our intuition, as the model without re-routing overestimates the delay for a passenger that misses a connection.
For the second and third case, the no-wait policy performs worst. In the second case, taking re-routing into account reduces the delay by 9.7 percent with respect to a no-wait policy and by 2.3 percent in comparison to the classical delay management model. In the third case, the reduction is 7.9 and 3.3 percent, respectively. The number of dropped connections is almost the same for the delay management models with and without explicit re-routing. However, it is about 30 percent less than with the no-wait policy. This shows that delaying only a few trains can improve the performance of the railway system as a whole. When re-routing is taken into account explicitly, slightly more connections are maintained. A possible explanation is that if a connection is dropped in the model without re-routing, the model with re-routing can maintain a connection to a later train by delaying that later train slightly. The classical model will not consider this possibility.

6 Conclusion and Further Research

In this paper, we introduced a model that allows to react to delayed trains not only by wait-depart decisions for the following trains but also by re-routing of passengers. For this purpose we introduced the origin and destination of the passengers as events in the event-activity network used in delay management and connected the wait-depart decisions to a shortest path problem in the resulting network. We proved that this problem is NP-hard. Furthermore, we developed an integer programming formulation for the delay management problem with re-routing. This novel formulation was tested on real-world instances of Netherlands Railways. We showed that improvements of 2-5% can be obtained by incorporating the re-routing aspect in the model. Furthermore, our model ensures that all passengers can reach their destination. Unfortunately, we were not able to solve very large real-world instances.
Therefore, we propose two directions of further research on delay management with re-routing. First, further special cases of the problem should be considered. For these special cases, faster solution procedures can be developed. For example, if the event-activity network has a special structure, this structure can be exploited to solve the delay management problem more efficiently. The methods to solve these easier problems can be used in heuristics to solve large-scale delay management problems in a short amount of time. Second, decomposing methods could be developed that split the problem in the wait-depart decisions on one hand and the re-routing of the passengers on the other hand.
In practice, the limited capacity of the infrastructure has a large impact on the real-time performance of a railway operator. Therefore, capacity constraints should also be integrated in the delay management model with re-routing that we presented in this paper.

References

A. Berger, R. Hoffmann, U. Lorenz, and S. Stiller. Online delay management: Pspace hardness and simulation. Technical Report ARRIVAL-TR-0097, ARRIVAL Project, 2007.

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

M. Garey and D. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

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: 31st International Workshop (WG 2005)*, volume 3787 of *Lecture Notes in Computer Science*, 2005.

M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. On-line delay management on a single train line. In *Algorithmic Methods for Railway Optimization*, number 4359 in 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.

L. D. 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.

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

L. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, L. Schrijver, A. Steenbeek, and R. Ybema. The New Dutch Timetable: The OR Revolution. *Interfaces*, 39:6–17, 2009.

C. Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42: 420–435, 2008.

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

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*, 2010. DOI 10.1287/trsc.1100.0318.

A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 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 *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes in Computer Science, pages 145–170. Springer, 2007.

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

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

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