

Georg-August-Universität Göttingen



Integer programming approaches for solving the delay management problem

Anita Schöbel

Preprint Nr. 2005-02

Preprint-Serie des
Instituts für Numerische und Angewandte Mathematik
Lotzestr. 16-18
D - 37083 Göttingen

Integer programming approaches for solving the delay management problem*

Anita Schöbel
Georg-August University, Göttingen

Mai 2005

Abstract

The *delay management problem* deals with reactions in case of delays in public transportation. More specifically, the aim is to decide if connecting vehicles should wait for delayed feeder vehicles or if it is better to depart on time. As objective we consider the convenience over all customers, expressed as the average delay of a customer when arriving at his or her destination.

We present path-based and activity-based integer programming models for the delay management problem and show the equivalence of these formulations. Based on these, we present a simplification of the (cubic) activity-based model which results in an integer *linear* program. We identify cases in which this linearization is correct, namely if the so-called *never-meet property* holds. Fortunately, this property is often almost satisfied in our real-world data. Finally, we show how to find an optimal solution in linear time if the never-meet property holds.

1 Introduction

A major reason for complaints about public transportation is the missing punctuality, which — unfortunately — is a fact in many transportation systems. Since it seems to be impossible to avoid delays completely, it is a necessary issue in the operative work of a public transportation company to deal with delayed vehicles. In this paper we focus on the convenience of the customers and present a model for minimizing the average delay over all passengers.

Let us consider some vehicle (e.g., a train g) that arrives at a station with a delay. At the station, there are other vehicles (e.g., buses h and h') ready to depart,

*This work was partially supported by Stiftung Innovation Rheinland-Pfalz

see Figure 1. What should each of these connecting vehicles do? There are two alternatives:

- A connecting vehicle h can **wait** to allow passengers to change from the delayed vehicle g to h .
- The connecting vehicle h can **depart** on time.

Unfortunately, both decisions have negative effects: In the first case, vehicle h causes a delay for passengers already within h , but also for customers who wish to board vehicle h later on, and possibly for subsequent other vehicles which will have to wait for its delay. In the second case, however, all customers who planned to change from the delayed vehicle g into h will miss their connection.

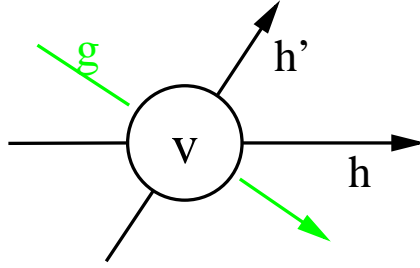


Figure 1: The wait-depart decision at one single station.

In the first case the connecting vehicle h does not depart at its scheduled time, but with a delay. The new departure time of h is called its *perturbed* timetable. In the second case, the *perturbed* departure time of h at v equals the scheduled one.

In case of some known delays, the *delay management problem* is to find wait-depart decisions and a perturbed timetable for all vehicles in the network, such that the sum of all delays over all customers is minimized. The delay of a customer is defined as the delay he has when he reaches his destination. Recently the NP-completeness of this problem has been shown (see [GJPS04]).

Since in the delay management problem new departure times for each vehicle at each station have to be determined, it is related to finding timetables in public transportation. In this field, a lot of research has been done for periodic and non-periodic timetables. An excellent overview on periodic timetabling is given by [Pee02]. We also refer to [Nac98, Car99, Gov98, vE01] and references therein. However, note the main difference between timetabling and delay management: In the timetabling problem the connections are given in advance, while in the delay management problem we have not only to find a (perturbed) timetable, but also to decide which connections should be maintained and which can be dropped.

How to *react* in case of delays has — due to the size and complexity of the problem — been mainly tackled by simulation and expert systems. We refer to [SBK01, SMBG01] for providing a knowledge-based expert system including a simulation of wait-depart decisions with a *what-if* analysis. Simulation has also been used in [Ack99, SM01].

In [GS02] the delay management problem has been formulated as a bicriterial problem, minimizing the number of missed connections and the delay of the vehicles simultaneously, and solved by methods of project planning. The weighted sum of these functions has been minimized in [RdVM98] by an enumeration procedure and a greedy heuristic within a max-plus algebraic model, see also [SvdB01]. Dynamic programming has been used in [GGJ⁺04] to identify polynomially solvable cases.

Integer programming formulations so far only exist as first attempts for the simple case without slack times, assuming that the customers on each edge are fixed (see the diploma theses of [Kli00] and [Sch01b]). In Section 4 we are able to identify cases in which such models are correct. A first *exact* linear integer model for the delay management problem is presented in [Sch01a], and will be reviewed in this paper in a more convenient notation at the beginning of Section 3.

Related work includes how to reduce delays by investing into new tracks ([EFK01, EF02]), how to minimize the sum of waiting times of customers at their starting stations in a stochastic context ([APW02]), and a first on-line model of the problem along a line ([GJPW04]).

The aim of this paper is to present a new and more general integer programming formulation of the delay management problem, for which we are able to develop solution approaches. Although our model can be applied to many different objective functions we specialize here on minimizing the sum of all delays over all customers. After introducing definitions and basic properties in Section 2 we present two different, but equivalent integer programming formulations for the delay management problem. In Section 4 we show that our new model can be linearized if a special condition, called *the never-meet property* holds. In Section 5 we show how to solve the delay management problem in linear time in this case. The paper is concluded by some remarks on future research.

2 Notation, concepts, and basic properties

We first introduce a new notation for the delay management problem, based on its representation as an *activity-on-arc project network* (see e.g. [Nac98] for using this concept in timetabling). As an example, a small event-activity network is depicted in Figure 2.

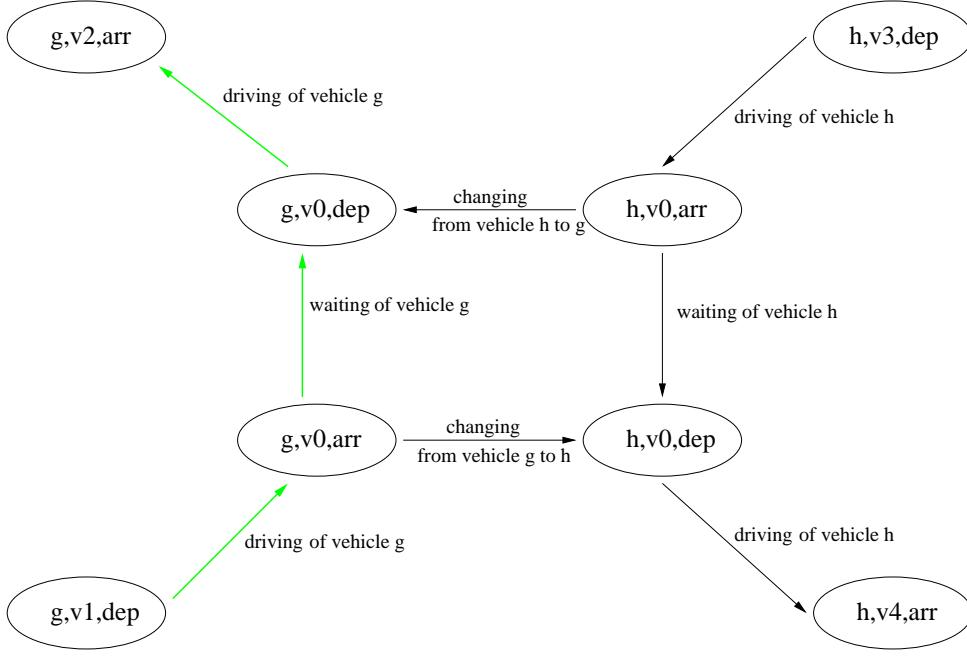


Figure 2: An event-activity network.

Notation 1 An arrival of a vehicle g at a station v is called an **arrival event** (g, v, arr) , while a **departure event** (g, v, dep) describes the departure of some vehicle g at some station v . The event activity network is a graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ where

- $\mathcal{E} = \mathcal{E}_{arr} \cup \mathcal{E}_{dep}$ is the set of all arrival and all departure events
- $\mathcal{A} = \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}_{change}$ is a set of directed arcs, called **activities**, defined by

$$\mathcal{A}_{wait} = \{((g, v, arr), (g, v, dep)) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep}\}$$

$$\mathcal{A}_{drive} = \{((g, v, dep), (g, u, arr)) \in \mathcal{E}_{dep} \times \mathcal{E}_{arr} : \text{vehicle } g \text{ goes directly from station } v \text{ to } u\},$$

$$\mathcal{A}_{change} = \{((g, v, arr), (h, v, dep)) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep} : \text{a changing possibility from vehicle } g \text{ into } h \text{ at station } v \text{ is required}\}.$$

The driving and waiting activities are performed by vehicles, while the changing activities are used by the customers. Note that \mathcal{N} is a special case of a time-expanded network and hence is acyclic. This means that a precedence relation \prec between events (or activities) is canonically given. $i \prec j$ hence indicates that there exists a path from i to j . We remark that for a given set of events (or of activities) a minimal element w.r.t. \prec always exists, but it needs not be unique.

Using the notation of event-activity networks, a *timetable* $\Pi \in \mathbb{Z}^{|\mathcal{E}|}$ is given by assigning a time Π_i (usually in minutes) to each event $i \in \mathcal{E}$ (see [Nac98]). Timetables are usually given in minutes and hence consist of integer values. The planned duration of activity $a = (i, j)$ is given by $\Pi_j - \Pi_i$. Furthermore, let $L_a \in \mathbb{N}$ be the minimal duration needed for performing activity a . We assume that the timetable is *feasible*, i.e.,

$$\Pi_j - \Pi_i \geq L_a \text{ for all } a = (i, j) \in \mathcal{A}.$$

We further assume that *source delays* are known at some of the events, where they might have occurred at the preceding activity or at the event itself. Let $\mathcal{E}_{del} \subseteq \mathcal{E}_{arr}$ denote the set of source-delayed events, and $d_i > 0$ indicate the delay they have. (For $i \notin \mathcal{E}_{del}$ the source delay $d_i = 0$.)

If source delays occur, some of the subsequent arrival and departure times Π_i can also not take place punctually, since the minimal durations L_a for subsequent activities have to be taken into account. The outcome $\Pi + y$ is called a *perturbed timetable*, and y_i is called the delay of event i . Such a perturbed timetable is feasible, if

- the source delays are taken into account, i.e., $\Pi_i + y_i \geq \Pi_i + d_i$, and
- the delay is carried over correctly from one event to the next, i.e.,
 $\Pi_j + y_j - (\Pi_i + y_i) \geq L_a$ holds for all driving and waiting activities $a = (i, j)$.

Defining the *slack time* s_a of an activity $a \in \mathcal{A}$ as the time which can be saved when performing activity a as fast as possible, i.e.,

$$s_a = \Pi_i - \Pi_j - L_a$$

we can equivalently restate the two above conditions in terms of the delay vector y as follows.

Definition 1 *A set of delays y_i for all $i \in \mathcal{E}$ is **feasible**, if*

$$y_i \geq d_i \text{ for all } i \in \mathcal{E} \text{ and} \tag{1}$$

$$y_i - y_j \leq s_a \text{ for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}. \tag{2}$$

Condition (2) makes sure that the delay at the start of activity a is transferred to its end, where it can be reduced by the slack time of a .

In the following we only use the slack times s and the delays y instead of the minimal durations L and the timetable Π .

Definition 1 only takes the driving and waiting activities into account. However, in the delay management problem the goal is to identify which changing activities

should be maintained and which ones can be dropped. For a changing activity we analogously require that

$$y_i - y_j \leq s_a \text{ if } a = (i, j) \text{ is maintained} \quad (3)$$

We are now in the position to specify feasible solutions of the delay management problem.

Definition 2 *A set of connections $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}$ together with a feasible set of delays y_i for all $i \in \mathcal{E}$ is a **feasible solution of the delay management problem**, if*

$$y_i - y_j \leq s_a \text{ for all } a = (i, j) \in \mathcal{A}^{fix},$$

i.e., for all connections $a \in \mathcal{A}^{fix}$ which are maintained.

Note that a timetable would also be feasible if some vehicles depart or arrive late without any reason. Such solutions are clearly not optimal, and hence we define a set of “most punctual” solutions.

Notation 2 *Let (\mathcal{A}^{fix}, y) be a feasible solution of the delay management problem. y is called **time-minimal** with respect to \mathcal{A}^{fix} if all feasible solutions (\mathcal{A}^{fix}, y') satisfy $y \leq y'$ (where as usual \leq is meant componentwise).*

It can be shown that there exists a unique time-minimal solution with respect to each set $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}$, which can be found efficiently by one of the following approaches. The proof of existence and uniqueness follows from the second approach.

Linear programming approach :

The following is an integer programming formulation whose solutions are exactly the time-minimal feasible solutions:

$$\min \sum_{i \in \mathcal{E}} y_i$$

$$\text{such that } y_i \geq d_i \text{ for all } i \in \mathcal{E}_{del} \quad (4)$$

$$y_i - y_j \leq s_a \text{ for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}^{fix} \quad (5)$$

$$y_i \in \mathbb{N} \text{ for all } i \in \mathcal{E}.$$

Since the resulting integer program has a totally unimodular coefficient matrix, the integrality condition $y \in \mathbb{N}^{|\mathcal{E}|}$ is not needed and the problem can be solved by linear programming.

Critical path method (CPM) :

The event-activity network can be transformed into a project network (as defined, e.g., in [Elm77]) by introducing one super-source s and taking

$$\mathcal{A}(\mathcal{A}^{fix}) = \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}^{fix}$$

and additional *timetable activities* $\{(s, i) : i \in \mathcal{E}\}$ as set of activities in the corresponding project network. The duration of an activity is set to L_a for $a \in \mathcal{A}$ and to the scheduled timetable Π_i if $a = (s, i)$. Then the earliest possible starting time of each activity is a time-minimal solution of the delay management problem. The following procedure uses the critical path method to determine the earliest starting times but is applied directly in the notation of slack times s and delays y according to Definition 1.

Algorithm 1: Calculating a time-minimal solution for a set \mathcal{A}^{fix}

Input: \mathcal{N} , d_i , s_a , \mathcal{A}^{fix} .

Output: Optimal (time-minimal) solution w.r.t. \mathcal{A}^{fix} .

Step 1. Sort $\mathcal{E} = \{i_1, \dots, i_{|\mathcal{E}|}\}$ according to \prec .

Step 2. For $k = 1, \dots, |\mathcal{E}|$: $y_{i_k} = \max\{d_{i_k}, \max_{a=(i, i_k) \in \mathcal{A}(\mathcal{A}^{fix})} y_i - s_a\}$

Step 3. Output: y_i , $i \in \mathcal{E}$

Longest path technique for the feasible differential problem :

It is also possible to transform the delay management problem with fixed connections to a feasible differential problem (defined, e.g., in [Roc84]). Note that the potential in this case is given by the delay for each node and the tension is the additional delay of each of the activities, see [Sch03] for details.

Using these observations, we can show the following basic result.

Lemma 1 *For some set $\mathcal{A}^{fix} \subseteq \mathcal{A}$ let $y(\mathcal{A}^{fix})$ denote a time-minimal solution w.r.t. \mathcal{A}^{fix} . Then*

1. $\mathcal{A}^1 \subseteq \mathcal{A}^2 \subseteq \mathcal{A}_{change} \implies y(\mathcal{A}^1) \leq y(\mathcal{A}^2)$.
2. $y = y(\mathcal{A}^{fix})$ satisfies $y_i \leq D = \max\{d_i : i \in \mathcal{E}\}$ for all $i \in \mathcal{E}$.

The first part of the lemma states, that the delays get smaller if connections are dropped, while the second part bounds the maximal delay of a single event in a time-minimal solution by the largest given source delay. Both results will be needed later.

Proof: The result can be shown by induction, using Algorithm 1 to calculate a time-minimal solution. To start, choose a minimal event i and note that $y_i = d_i \leq D$ and this is independent of the set of fixed connections chosen. Now take any event $j \in E$. From the induction assumption we may assume that $y_i \leq D$ and $y_i(\mathcal{A}^1) \leq y_i(\mathcal{A}^2)$ holds for any predecessors i (w.r.t. \prec) of j . From (2) we obtain $y_j \leq D$ and

$$\begin{aligned} y_j(\mathcal{A}^1) &= \max\{d_j, \max_{a=(i,j) \in \mathcal{A}(\mathcal{A}^1)} y_i(\mathcal{A}^1) - s_a\} \\ &\leq \max\{d_j, \max_{a=(i,j) \in \mathcal{A}(\mathcal{A}^2)} y_i(\mathcal{A}^2) - s_a\} = y_j(\mathcal{A}^2). \end{aligned}$$

QED

As mentioned before, our objective is to minimize the sum of all delays over all customers. To this end, we have to specify the customers data.

A customer's paths is given as a sequence of events, i.e.,

$$p = (i_1, i_2, \dots, i_{p_L})$$

where $i_k \in \mathcal{E}$ are events, and $(i_k, i_{k+1}) \in \mathcal{A}$ are activities. We will write $a = (i_k, i_{k+1}) \in p$ in this case. Note that i_1 is a departure event, i_2 an arrival event, $i_3 \in \mathcal{E}_{dep}$ and so on. Furthermore, $i(p)$ denotes the last event on path p and w_p the number of passengers who want to use path p . We denote \mathcal{P} as the set of all customers' paths.

To calculate the delay of a passenger on path p we need the following two basic assumptions:

1. There is one (common) time period T for all vehicles.
2. In the next time period all vehicles are on time.

In praxis, both assumptions are usually not satisfied. The first of them can be relaxed a bit, allowing different periods for each of the activities. Taking the largest of the periods of all lines overestimates the delay, but seems to be a reasonable approach. The second assumption is accepted by practitioners since the planning period in on-line disposition is usually less than the time period T (often one hour). It is an open problem to deal with future delays by using stochastic optimization.

To calculate the delay of a customer using some path $p \in \mathcal{P}$, we have to distinguish the following two cases.

Case 1: If all connections of path p are maintained (i.e., the path is *maintained*), the delay of a passenger on path p is the arrival delay $y_{i(p)}$ of his last event $i(p)$.

Case 2: If at least one connection of path p is missed, the delay of a passenger on path p is given by T .

We are finally in the position to define the **(total) delay management problem**.

(TDM): Given $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, slack times s_a for all $a \in \mathcal{A}$, source delays $d_i, i \in \mathcal{E}$ and a set of weighted paths \mathcal{P} , find a feasible pair $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}$ with delays $y_i, i \in \mathcal{E}$ such that the sum of all delays over all customers is minimal.

3 Two models for delay management

In this section we first review the formulation of [Sch01a] in the new notation introduced above. In contrast to this path-based model we then present a new activity-based formulation and show that both models are equivalent.

3.1 Path-based formulation

As first model we present a path-oriented description of (TDM) (based on the formulation in [Sch01a]) which uses the following variables

$$z_p = \begin{cases} 0 & \text{if all connections on path } p \text{ are maintained} \\ 1 & \text{otherwise} \end{cases}$$

(TDM-A)

$$\min f_{\text{TDM-A}} = \sum_{p \in \mathcal{P}} w_p (y_{i(p)} (1 - z_p) + T z_p)$$

such that

$$y_i \geq d_i \text{ for all } i \in \mathcal{E}_{del} \quad (6)$$

$$y_i - y_j \leq s_a \text{ for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \quad (7)$$

$$-M z_p + y_i - y_j \leq s_a \text{ for all } p \in \mathcal{P}, a = (i, j) \in p \cap \mathcal{A}_{change} \quad (8)$$

$$y_i \in \mathbb{N} \text{ for all } i \in \mathcal{E} \quad (9)$$

$$z_p \in \{0, 1\} \text{ for all } p \in \mathcal{P} \quad (10)$$

The first two constraints (6) and (7) are the same as (1) and (2). Constraint (8) makes sure that all connections on a maintained path (i.e. a path with $z_a = 0$)

satisfy (3). Finally, the objective function sums up the delay according to the two cases mentioned on page 8.

In the following, we always assume $M \geq D = \max\{d_i : i \in \mathcal{E}\}$.

The given formulation of model (TDM-A) can be linearized by substituting the quadratic terms $y_{i(p)}(1 - z_p)$ by additional variables q_p , leading to the following model (TDM-B).

(TDM-B)

$$\min f_{\text{TDM-B}} = \sum_{p \in \mathcal{P}} w_p(q_p + Tz_p)$$

such that (6) – (10) hold, and such that

$$-Mz_p + y_{i(p)} - q_p \leq 0 \quad \text{for all } p \in \mathcal{P} \quad (11)$$

$$q_p \geq 0 \quad \text{for all } p \in \mathcal{P} \quad (12)$$

Lemma 2 *The linearization is correct.*

Proof:

(TDM-A) \implies (TDM-B): Let (y, z) be a feasible solution of (TDM-A). Due to Lemma 1 we may assume that $y_i \leq D$ for all $i \in \mathcal{E}$. For all $p \in \mathcal{P}$ define $q_p = y_{i(p)}(1 - z_p)$. Since $y_{i(p)} \leq D \leq M$ we get for all $p \in \mathcal{P}$ that

$$-Mz_p + y_{i(p)} \leq -y_{i(p)}z_p + y_{i(p)} = q_p.$$

Hence, (y, z, q) is feasible for (TDM-B), and both solutions have the same objective value.

(TDM-B) \implies (TDM-A): Let (y, z, q) be a feasible solution of (TDM-B). Then (y, z) is also feasible for (TDM-A). From (11) and (12) we conclude that

$$\begin{aligned} q_p &\geq y_{i(p)} \text{ if } z_p = 0 \\ q_p &\geq 0 \text{ if } z_p = 1. \end{aligned}$$

Consequently, $q_p \geq y_{i(p)}(1 - z_p)$, i.e., $f_{\text{TDM-A}} \leq f_{\text{TDM-B}}$.

QED

Note that the linear formulation is significantly weaker as the quadratic formulation (TDM-A). This is due to the fact, that the feasible set of the linear programming relaxation increased. More intuitively, one would like to use variables \bar{z}_a determining if a connection $a \in \mathcal{A}_{\text{change}}$ should be maintained or not. This yields a stronger activity-based formulation for (TDM) which is derived in the next section.

3.2 Activity-based formulation

In this model we use variables for each changing activity \bar{z}_a describing if connection $a \in \mathcal{A}_{change}$ is missed ($\bar{z}_a = 1$) or maintained ($\bar{z}_a = 0$). The idea of the activity-based formulation is to calculate the total delay by summing up the *additional delays* over all activities $a \in \mathcal{A}$.

To this end, let us first consider some activity $a \in \mathcal{A} \setminus \mathcal{A}_{change}$. We want to calculate the additional delay customers will get while using this activity. The delay customers already have at the start of $a = (i, j)$ is y_i , and at the end of a their delay is y_j . This means, the tension $y_j - y_i$ is the additional delay gained by the customers while performing activity a . Note that this additional delay can be negative, meaning that slack times are used to compensate an already existing delay.

For changing activities we have to be more careful. Let $a = (i, j) \in \mathcal{A}_{change}$ and suppose first that a is maintained. Then the additional delay on a is again the tension $y_j - y_i$. On the other hand, if a is missed, the additional delay for the customers who planned to use activity a is given by $T - y_i = y_j - y_i + T - y_j$, since they now have to wait the remaining time period until the next (non-delayed) vehicle arrives for carrying on their journey.

We further need to extend the event-activity network by defining

$$\begin{aligned}\mathcal{E}^s &= \mathcal{E} \cup \{s\} \\ \mathcal{A}^s &= \mathcal{A} \cup \{(s, i) : i \in \mathcal{E}_{dep}\} \text{ and} \\ \mathcal{P}^s &= \{(s, i_1^p, \dots, i_L^p) : p \in \mathcal{P}\}.\end{aligned}$$

The additional event s represents the arrival of the customers at their first station (by foot or by a means of transport which is not considered in the delay management problem). The extension makes sure that the delay of a customer waiting at some station for his first (delayed) vehicle to come, is taken into account. We always assume that customers reach their first station without any delay, i.e.,

$$y_s = 0.$$

Now we can present the new model. As before, we assume that $T, M \geq D$.

The following additional variables are necessary for (TDM-C).

$$\begin{aligned}\bar{z}_a^p &= \begin{cases} 1 & \text{if activity } a \text{ is reached on path } p \text{ without any missed} \\ & \text{connection before} \\ 0 & \text{otherwise} \end{cases} \\ w_a &= \text{number of customers who really use activity } a\end{aligned}$$

We stress that the number of customers w_a (really) using activity $a \in \mathcal{A}$ is a variable, since it depends on the wait-depart decisions whether customers using a path $p \in \mathcal{P}^s$ will reach all activities $a \in \mathcal{P}$ or not.

(TDM-C)

$$\min f_{\text{TDM-C}} = \sum_{a=(i,j) \in \mathcal{A}^s} w_a(y_j - y_i) + \sum_{a=(i,j) \in \mathcal{A}_{\text{change}}} w_a \bar{z}_a(T - y_j)$$

such that

$$y_i \geq d_i \quad \text{for all } i \in \mathcal{E}_{\text{del}} \quad (13)$$

$$y_i - y_j \leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}} \quad (14)$$

$$-M\bar{z}_a + y_i - y_j \leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{\text{change}} \quad (15)$$

$$\tilde{z}_a^p + \sum_{\substack{\tilde{a} \in p \cap \mathcal{A}_{\text{change}}: \\ \tilde{a} \prec a}} \bar{z}_{\tilde{a}} \geq 1 \quad \text{for all } p \in \mathcal{P}^s \text{ and } a \in p \quad (16)$$

$$\begin{aligned} \tilde{z}_a^p + \bar{z}_{\tilde{a}} &\leq 1 \quad \text{for all } p \in \mathcal{P}^s \text{ and for all } a, \tilde{a} \in p \\ &\quad \text{with } \tilde{a} \in \mathcal{A}_{\text{change}} \text{ and } \tilde{a} \prec a \end{aligned} \quad (17)$$

$$w_a = \sum_{p \in \mathcal{P}^s: a \in p} w_p \tilde{z}_a^p \quad \text{for all } a \in \mathcal{A}^s \quad (18)$$

$$y_i \in \mathbb{N} \quad \text{for all } i \in \mathcal{E} \quad (19)$$

$$\bar{z}_a \in \{0, 1\} \quad \text{for all } a \in \mathcal{A}^s \quad (20)$$

$$\tilde{z}_a^p \in \{0, 1\} \quad \text{for all } p \in \mathcal{P}^s, a \in \mathcal{A}^s \quad (21)$$

$$w_a \in \mathbb{N} \quad \text{for all } a \in \mathcal{A}^s \quad (22)$$

In the objective function the additional amount of delay on each activity is multiplied by the number of customers *really* using it. Restrictions (13) and (14) again correspond to (1) and (2), while (15) models that (3) has to be satisfied exactly for maintained connections, i.e. connections a with $\bar{z}_a = 0$. Restriction (16) defines the values of \tilde{z}_a^p such that they are forced to be 1, if no connection on path p before a has been missed, and (17) makes sure that $\tilde{z}_a^p = 0$ for all activities a after a missed connection \tilde{a} on path p . Finally, (18) determines the number of customers really using activity a .

Note that for technical reasons we need to be able to extend any feasible solution $y_i, i \in \mathcal{E}$ to a feasible solution $(y, C(y)) := (y, \bar{z}(y), \tilde{z}(\bar{z}), w(\tilde{z}))$ of (TDM-C), where $(y, C(y))$ yields the same or a better objective function value for (TDM-C). This is done as follows.

$$\bar{z}_a(y) = \begin{cases} 0 & \text{if } y_i - y_j \leq s_a \\ 1 & \text{otherwise} \end{cases} \quad \text{for all } a = (i, j) \in \mathcal{A}_{\text{change}}, \quad (23)$$

$$\tilde{z}_a^p(\bar{z}) = \max \left\{ 1 - \sum_{\substack{\tilde{a} \in p \cap \mathcal{A}_{\text{change}}: \\ \tilde{a} \prec a}} \bar{z}_{\tilde{a}}, 0 \right\} \quad \text{for all } p \in \mathcal{P}^s, a \in p, \quad (24)$$

$$w_a(\tilde{z}) = \sum_{p \in \mathcal{P}^s: a \in p} w_p \tilde{z}_a^p \quad \text{for all } a \in \mathcal{A}^s. \quad (25)$$

3.3 Relation between the models

Theorem 1 *(TDM-A) and (TDM-C) are equivalent. In particular, both models lead to the same set of optimal solutions $y \in \mathbb{R}^{|\mathcal{E}|}$.*

Proof: see Appendix.

On a first glance, (TDM-C) does not seem to be useful for solving the delay management problem better than (TDM-A), since (TDM-A) can be linearized (see Lemma 2) while (TDM-C) is cubic. Moreover, (TDM-C) is much larger in terms of variables, constraints, and non-zero entries of the coefficient matrix. However, it has some advantages. First, it is more general since it allows to replace the common time period T by time periods T_a for each changing activity $a \in \mathcal{A}_{change}$, which is a step to more realistic models and to relaxing our first assumption on page 8. Secondly, as the proof in the appendix shows, (TDM-B) is a stronger formulation than (TDM-A), since the decision variables \bar{z}_a allow less freedom than the decision variables z_p . Hence, e.g., a classical branch-and-bound procedure using the variables \bar{z}_a for branching can be easily implemented for (TDM-C) while for (TDM-A) other methods, e.g., constraint branching have to be investigated. Last, we will be able to utilize (TDM-C) to present a linear-time algorithm which solves the delay management problem exactly for a special class of problems.

4 Constant weights and the never-meet property

In order to solve (TDM-C) we fix the weights w_a as parameters instead of calculating them during the optimization. Doing so, we obtain the *total delay management problem with constant weights*. Its formulation is given by deleting constraints (16), (17), and (18) in (TDM-C), and fixing

$$w_a = \sum_{p \in \mathcal{P}^s : a \in p} w_p \text{ for all } a \in \mathcal{A}^s \quad (26)$$

as parameters, i.e., setting w_a as the planned “traffic load” on activity a . We obtain:

$$\min f_{\text{TDM-const}'} = \sum_{a=(i,j) \in \mathcal{A}^s} w_a (y_j - y_i) + \sum_{a=(i,j) \in \mathcal{A}_{change}} w_a \bar{z}_a (T - y_j)$$

such that (13),(14),(15),(19), and (20) hold.

We can further rewrite $f_{\text{TDM-const}'}$ as follows. For $i \in \mathcal{E}$ let

$$w_i = \sum_{p \in \mathcal{P} : i(p)=i} w_p \quad (27)$$

be the number of customers with final destination i . Since

$$\begin{aligned}
\sum_{a=(i,j) \in \mathcal{A}^s} w_a(y_j - y_i) &= \sum_{p \in \mathcal{P}^s} w_p \sum_{a=(i,j) \in p} y_j - y_i \\
&= \sum_{p \in \mathcal{P}} w_p (y_{i(p)} - y_s) \\
&= \sum_{i \in \mathcal{E}} \sum_{\substack{p \in \mathcal{P}: \\ i(p)=i}} w_p y_i = \sum_{i \in \mathcal{E}} w_i y_i
\end{aligned} \tag{28}$$

we rewrite

$$f_{\text{TDM-const}'} = \sum_{i \in \mathcal{E}} w_i y_i + \sum_{a=(i,j) \in \mathcal{A}_{\text{change}}} w_a \bar{z}_a (T - y_j).$$

We show that in general, we make a mistake by fixing the weights as above, which has not been realized in many attempts or simulation approaches for the delay management problem.

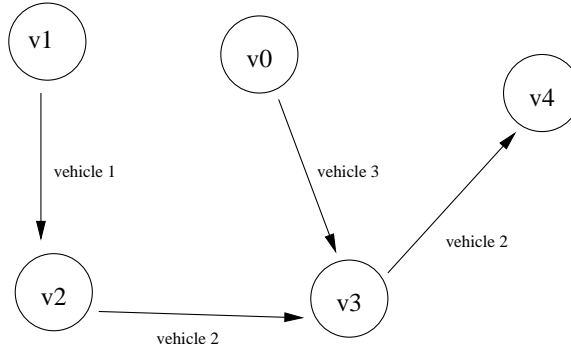


Figure 3: An example in which fixing the weights is not correct.

We assume there are three vehicles 1, 2, and 3, where vehicle 1 and vehicle 3 reach the stations v_2 and v_3 with a delay, see Figure 3. We consider a customers' path $p = (v_1, v_2, v_3, v_4)$ using vehicle 1 until station v_2 , changing to vehicle 2 and passing via v_3 to its destination v_4 . Suppose that vehicle 2 is not waiting for vehicle 1 at station v_2 , such that the path p is not maintained. Assume further that vehicle 2 waits for the delayed vehicle 3 at station v_3 . If we have not adapted the weights, the customers on path p are counted twice in the objective function: First, since they missed their connection at station v_2 , and secondly, since they reach their final destination v_4 with a delay. This double counting can in general lead to wrong decisions.

Fortunately, there are problem instances for which the model with constant weights is correct, apart from the trivial case in which no customer changes at all. For example, it can be shown that the model with constant weights is correct,

if we only allow paths of the form $p = (i_1, i_2, \dots, i_{L-2}, i_{L-1}, i_L)$ where p contains at most one changing activity (i_{L-2}, i_{L-1}) followed by not more than one driving activity, see [Sch03]. A more interesting case, in which we make no mistake by using the constant weights will be described next.

Since $f_{\text{TDM-const}'}$ still is no linear function we further simplify the model. In the following we simply forget about subtracting y_j in the second part of the objective, to obtain the **linear** program (**TDM-const**).

$$\min f_{\text{TDM-const}} = \sum_{i \in \mathcal{E}} w_i y_i + \sum_{a \in \mathcal{A}_{\text{change}}} w_a T \bar{z}_a$$

such that

$$y_i \geq d_i \quad \text{for all } i \in \mathcal{E}_{\text{del}} \quad (29)$$

$$y_i - y_j \leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}} \quad (30)$$

$$-M \bar{z}_a + y_i - y_j \leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{\text{change}} \quad (31)$$

$$y_i \in \mathbb{N} \quad \forall i \in \mathcal{E}$$

$$\bar{z}_a \in \{0, 1\} \quad \text{for all } a \in \mathcal{A}_{\text{change}}$$

Each feasible solution of (TDM-const) yields an upper bound on (TDM). But the main advantage of (TDM-const) is due to the surprising fact that (TDM-const) is equivalent to (TDM) in a large class of practical instances. To this end, we need some further technical details.

Notation 3 $\mathcal{H}(i) = \{j \in \mathcal{E} : \text{there exists a (directed) path from } i \text{ to } j\}$.

Furthermore, let $\mathcal{N}(i)$ be the subgraph consisting of the events in $\mathcal{H}(i)$ and their induced activities.

Note that for all $j \in \mathcal{H}(i)$ we have $i \preceq j$.

Definition 3 The delay management problem has the **never-meet property** if

1. $\mathcal{N}(i)$ is a tree for all $i \in \mathcal{E}_{\text{del}}$, and
2. $\mathcal{H}(i) \cap \mathcal{H}(j) = \emptyset$ for all $i, j \in \mathcal{E}_{\text{del}}$ with $i \neq j$,

i.e., $\bigcup_{i \in \mathcal{E}_{\text{del}}} \mathcal{N}(i)$ is a vertex-disjoint out-forest.

The interpretation of the never-meet property is the following: By calculating the time-minimal solution with respect to some given set \mathcal{A}^{fix} , but without using slack-times, we can find out how far the source delays can spread out in this solution in the worst case. The never-meet property requires that in **no** feasible solution of (TDM) two delayed vehicles or paths will meet. Note that the

formulation includes that source delays can only occur after non-delayed events.

If the never-meet property holds, we will show the following: In every time-minimal solution all events following a non-maintained connection are punctual, and all changing activities following a non-maintained connection are maintained. This property will be important for proving Theorem 2.

Lemma 3 *Let (TDM) have the never-meet property and let $(y, C(y))$ be a feasible (time-minimal) solution of (TDM-C). Let $\tilde{a} = (\tilde{i}, \tilde{j}) \in \mathcal{A}_{change}$. If $\bar{z}_{\tilde{a}} = 1$ (i.e. \tilde{a} is not maintained) we have the following.*

1. $y_i = 0$ for all $i \in \mathcal{H}(\tilde{j})$, i.e. all events following \tilde{j} are on time, and
2. $\bar{z}_a = 0$ for all $a = (i, j)$ with $i \in \mathcal{H}(\tilde{j})$, i.e., all connections following \tilde{j} are maintained.

Proof: From $\bar{z}_{\tilde{a}} = 1$ we know from (23) that $y_{\tilde{i}} > 0$. Hence there exists a source-delayed event $i_1 \in \mathcal{E}_{del}$ such that $\tilde{i} \in \mathcal{H}(i_1)$.

Now suppose there exists $i \in \mathcal{H}(\tilde{j}) \subseteq \mathcal{H}(i_1)$ with $y_i > 0$. Since $\bar{z}_{\tilde{a}} = 1$ the delay of i is not caused by d_{i_1} , but by another source-delayed event $i_2 \in \mathcal{E}$. But this means that $\mathcal{H}(i_1) \cup \mathcal{H}(i_2) \neq \emptyset$, a contradiction to the never-meet property.

Finally, consider $a = (i, j)$ with $i, j \in \mathcal{H}(\tilde{j})$. From part 1 we know that $y_i = y_j = 0$, hence (23) yields $\bar{z}_a = 0$.

QED

Theorem 2 *Model (TDM-const) is correct if the never-meet property holds.*

Proof: We show that (TDM-C) and (TDM-const) are equivalent in this case.

Clearly, a feasible solution (y, \bar{z}) of (TDM-const) can be extended to a feasible solution $(y, C(y))$ of (TDM-C) with equal or better objective value, see (23), (24), and (25).

The other direction is the interesting one: We show that each feasible solution of (TDM-C) corresponds to a feasible solution of (TDM-const) with the same or better objective value. More precisely, given some feasible solution of (TDM-C) with delay y , let $(y, C(y)) = (y, \bar{z}, \tilde{z}, w^{real})$ denote a (maybe better) feasible solution of (TDM-C). We show that (y, \bar{z}) is a feasible solution of (TDM-const) with the same objective value as $(y, C(y))$.

Feasibility of y, \bar{z} for (TDM-const) is trivially satisfied. It remains to show that

$$f_{\text{TDM-C}}(y, \bar{z}, \tilde{z}, w) = f_{\text{TDM-const}}(y, \bar{z}).$$

To this end, suppose that for some $\bar{a} = (\bar{i}, \bar{j}) \in \mathcal{A}$ we made a mistake by fixing the weights, i.e., the number of customers $w_{\bar{a}}$ who planned to use \bar{a} does not equal the

number of customers $w_{\bar{a}}^{real}$, really using \bar{a} . To compare the objective functions of (TDM-const) and (TDM-C) we replacing the first term of (TDM-const) by equation (28) and see that in this case it suffices to show that

$$y_{\bar{j}} - y_{\bar{i}} = 0,$$

and that, if $\bar{a} \in \mathcal{A}_{change}$

$$z_{\bar{a}} = 0,$$

This means that the error we make by using the wrong weights does not influence the value of the objective function.

From $w_{\bar{a}} \neq w_{\bar{a}}^{real}$ we get (by comparing (25) and (26)), that

$$\sum_{p \in \mathcal{P}^s: \bar{a} \in p} w_p = w_{\bar{a}} \neq w_{\bar{a}}^{real} = \sum_{p \in \mathcal{P}^s: \bar{a} \in p} w_p \tilde{z}_{\bar{a}}^p.$$

Hence there exists some path $p \in \mathcal{P}$ containing \bar{a} such that $\tilde{z}_{\bar{a}}^p = 0$. Due to (24) there exists $\tilde{a} \in p$ with $\tilde{a} \prec \bar{a}$ and $\tilde{z}_{\tilde{a}} = 1$. Without loss of generality let us take $\tilde{a} = (\tilde{i}, \tilde{j})$ minimal with this property, i.e., we choose the first changing activity on path p that is marked as missed. For an illustration, see Figure 4.



Figure 4: The path p in the proof of Theorem 2. The grey events belong to $\mathcal{H}(\tilde{j})$.

Since $\bar{i}, \bar{j} \in \mathcal{H}(\tilde{j})$ we derive from Lemma 3 that

- $y_{\bar{i}} = y_{\bar{j}} = 0$, and
- if $\bar{a} \in \mathcal{A}_{change}$ then $\bar{z}_{\bar{a}} = 0$.

Hence, $y_{\bar{j}} - y_{\bar{i}} = 0$, and if $\bar{a} \in \mathcal{A}_{change}$ we have that $z_{\bar{a}} = 0$, which completes the proof.

QED

We tested the never-meet property in practice, which can be done efficiently by the forward phase of the critical path method (with zero slack times and $\mathcal{A}^{fix} = \mathcal{A}_{change}$). As data we used a part of the public transportation network of Rheinland-Pfalz, Germany. The data consists of 823 stations, linked by 2118 edges. Since the never-meet property does not depend on the set of paths \mathcal{P} , we used four different sets \mathcal{U}_5 , \mathcal{U}_{10} , \mathcal{U}_{30} , and \mathcal{U}_{60} of connections, where set \mathcal{U}_x contains reasonable connections with a scheduled waiting time of less than x

minutes. By “reasonable” we mean that we do not consider connections where a transfer results in going directly back to the previous station. The sizes of the sets U_x range from 6531 (for \mathcal{U}_5) up to 80716 (for \mathcal{U}_{60}). The resulting event-activity network has a size of 46720 events. The number of activities depends on the set U_x used and varies between 51937 and 126122.

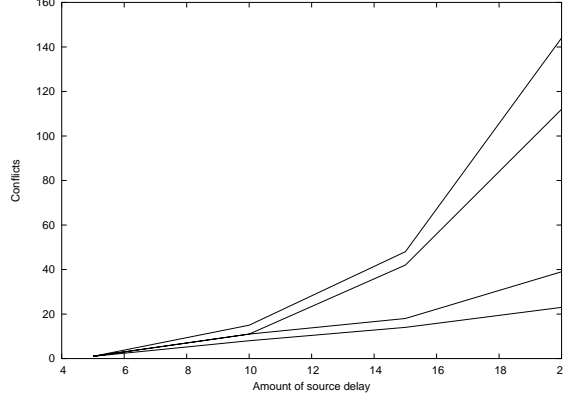


Figure 5: Conflicts with the never-meet property as function of the source delay, if 10 vehicles are delayed.

In our analysis we counted the number of activities which have to be deleted such that the never-meet property holds. The results are shown in Figures 5 and 6. In both figures the depicted functions correspond to the different sets of relevant connections. The lowest function uses \mathcal{U}_5 as set of connections, the next function corresponds to \mathcal{U}_{10} , then \mathcal{U}_{30} , and the top function refers to \mathcal{U}_{60} , confirming that the number of conflicts with the never-meet property increases if the set of connections is enlarged. Figure 5 shows the number of conflicts with the never-meet property as a function of the source delay, if we assume that 10 vehicles are delayed. It turns out that we can expect less than 50 conflicts if the source delays are smaller than 15 minutes.

In Figure 6 the number of conflicts with the never-meet property is depicted as a function of the number of delayed vehicles. This figure is calculated with a source delay of 15 minutes. Again, it turns out that not more than 50 conflicts are likely if the number of delayed vehicles is smaller than 10.

The reason for the relatively low number of conflicts in practice is in particular due to the fact that we only consider events that can gain a delay. We observed that most conflicts with the never-meet property arise at events within the city traffic, while the never-meet property is more likely to hold for transportation systems in a rural environment.

But all this is only helpful if we can draw advantage of the simplified model with constant weights in terms of efficiently solving it. As a first justification, our

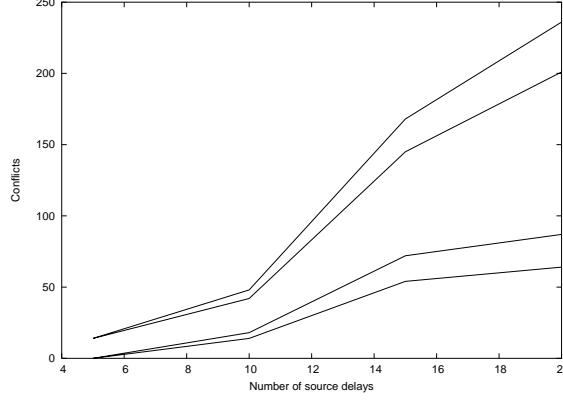


Figure 6: Conflicts with the never-meet property as function of the number of delayed vehicles, assuming a source delay of 15 minutes for each vehicle.

numerical results with Xpress indicate that the integer problem can be solved nearly in the same time as its relaxation when the problem instance is close to the never-meet property. In the next section this behavior will become clear when we discuss how to solve (TDM-const) if the never-meet property holds.

5 Solving (TDM) in the case of the never-meet property

The main result of this section is a linear-time algorithm for solving the delay management problem exactly if the never-meet property holds. We first remark that preprocessing should be done before using any of the approaches below. In such a preprocessing step we can delete all parts of the event-activity network where the delay can never spread (e.g. due to large slack times before). These events can be identified by using Algorithm 1 with $A^{fix} = \mathcal{A}_{change}$. In the following we hence always assume that $\mathcal{E} = \mathcal{E}_{reduced} = \{i \in \mathcal{E} : y_i(\mathcal{A}_{change}) > 0\}$. Note that in real-world instances, this type of reduction often leads to significantly smaller networks.

5.1 The special case with zero slack times

First, let us consider the special case (TDM-const-zero), in which

- all source delays have the same amount, i.e., $d_i \in \{0, D\}$ for all $i \in \mathcal{E}$, and
- all slack times are equal to zero, i.e., $s_a = 0$ for all $a \in \mathcal{A}$.

Let y be a time-minimal solution of this problem. Then $y_i \in \{0, D\}$ for all $i \in \mathcal{E}$. This means that we can use binary variables y_i instead of integer ones, with

$$y_i = \begin{cases} 1 & \text{if event } i \text{ is delayed by } D \\ 0 & \text{if event } i \text{ is not delayed.} \end{cases}$$

Consequently, $M = 1$ is large enough and (TDM-const) — even with the first objective $f_{\text{TDM-const}'}$ introduced on page 13 — simplifies to the following **linear** program. Recall that $\mathcal{E}_{del} = \{i \in \mathcal{E} : d_i > 0\}$.

(TDM-const-zero)

$$\min \sum_{a=(i,j) \in \mathcal{A}^s} w_a D(y_j - y_i) + \sum_{a=(i,j) \in \mathcal{A}_{change}} w_a \bar{z}_a (T - D)$$

such that

$$-y_i \leq -1 \quad \text{for all } i \in \mathcal{E}_{del} \quad (32)$$

$$y_i - y_j \leq 0 \quad \text{for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \quad (33)$$

$$\bar{z}_a + y_i - y_j \leq 0 \quad \text{for all } a = (i, j) \in \mathcal{A}_{change} \quad (34)$$

$$y_i \in \{0, 1\} \quad \forall i \in \mathcal{E}$$

$$\bar{z}_a \in \{0, 1\} \quad \forall a \in \mathcal{A}_{change},$$

where $w_a = \sum_{p \in \mathcal{P}^s : a \in p} w_p$ for all $a \in \mathcal{A}^s$ are given parameters as before (see, e.g., (26)).

Theorem 3 *(TDM-const-zero) can be solved in polynomial time.*

Proof: Let $C = |\mathcal{A}_{change}|$, $\bar{C} = |\mathcal{A}_{drive} \cup \mathcal{A}_{wait}|$ and $\bar{D} = |\mathcal{E}_{del}|$. Moreover, let I_K denote the unit matrix of size $K \times K$ and $O_{K,L}$ the zero matrix of size $K \times L$. Then the coefficient matrix of (TDM-const-zero) is

$$\Phi = \left(\begin{array}{c|c} -I_{\bar{D}} & 0_{\bar{D},C} \\ \hline \Theta^T & 0_{\bar{C},C} \\ & \hline & I_C \end{array} \right),$$

where the $|\mathcal{A}| \times |\mathcal{E}|$ -matrix Θ^T is the transposed of the node-arc-incidence matrix Θ of \mathcal{N} , and hence totally unimodular. Consequently, Φ also is totally unimodular. Hence the LP-relaxation of (TDM-const-zero) can be used to solve the problem in polynomial time by linear programming methods, (see e.g., [NW88]).

QED

Note that (TDM-const-zero) is equivalent to the models developed independently in diploma theses by Kliwer [Kli00] and Scholl [Sch01b], where the latter author also recognized the total unimodularity of the model.

In the case of non-zero slack times, the objective value of (TDM-const-zero) is still an upper bound on the objective value of (TDM-const).

5.2 Allowing arbitrary slack times

Since we assume that the never-meet property holds, we deal with the formulation (TDM-const) and show that this problem can be solved efficiently in $O(|\mathcal{A}|)$ time. The reason for this consists of the two facts listed below.

- First, if we fix $\bar{z}_a = 1$ for some $a = (i, j)$, we can set $y_{i'} = 0$ for all $i' \in \mathcal{H}(j)$ and know that all subsequent connections are maintained (Lemma 3).
- Secondly, the problem can be decomposed into at most $|\mathcal{A}_{change}|$ independent subproblems due to the following lemma, which follows directly from the never-meet property.

Lemma 4 *Let $i, j \in \mathcal{E}$, $i \neq j$, and let (y, \bar{z}) be a feasible solution of (TDM-const) with $y_i > 0, y_j > 0$. If the never-meet property holds, exactly one of the following three cases occurs.*

$$\mathcal{H}(i) \subseteq \mathcal{H}(j) \quad \text{or} \quad \mathcal{H}(j) \subseteq \mathcal{H}(i) \quad \text{or} \quad \mathcal{H}(i) \cap \mathcal{H}(j) = \emptyset.$$

The idea of the algorithm is to decompose the problem iteratively into subproblems, and solve them bottom-up. A subproblem P_a is identified by a changing activity $a = (i, j)$ and represents the delay management problem on the subgraph $\mathcal{N}(i)$ (recall Notation 3 on page 15) with a single source delay at event i . P_a might be decomposable into subproblems itself. Formally, we define

$$\text{SP}(a) = \{a' \in \mathcal{A}_{change} : \text{there exists a directed path from } a \text{ to } a' \text{ not containing any other changing activity}\}$$

The subproblems of the problem itself are collected in $\text{SP}(a_0)$, and can be derived by taking all changing activities reachable directly from one of the source-delayed events.

We remark that all subproblems within the same set $\text{SP}(a)$ are independent of each other due to the never-meet property.

In Algorithm 2, subproblems that might further be decomposed are stored in “Decompose”, and if a subproblem cannot be decomposed any more it is collected in “Compose”. Moreover, at the end of Step 2 of the algorithm, for each subproblem identified by some changing activity a ,

- $\text{maintain}(a)$ contains the value of the objective function of the subproblem if a is maintained, and
- $\text{miss}(a)$ contains the objective value if a is missed.
- $f(a)$ contains the minimum of $\text{maintain}(a)$ and $\text{miss}(a)$.

To compute $\text{maintain}(a)$ we need to calculate the minimum delay which occurs if a is maintained. in contrast to $\mathcal{H}(i)$ which is the set of events that can be reached from i , if *all* $a \in \mathcal{A}_{\text{change}}$ can be used we now define

$$\mathcal{G}(i) = \{j \in \mathcal{E} : \text{there exists a path from } i \text{ to } j \text{ with activities in } \mathcal{A}_{\text{wait}} \cup \mathcal{A}_{\text{drive}}\}$$

as the set of events that can be reached from i without passing any changing activity. Furthermore, assume that $i \in \mathcal{E}$ has a delay $d_i > 0$, and let y be a time-minimal solution. The minimum delay that will be caused by d_i independent of any wait-depart decision is then given by

$$G(i, d_i) = \sum_{j \in \mathcal{G}(i)} w_j y_j.$$

The algorithm can now be stated.

Algorithm 2: Enumeration for (TDM-const)

Input: $\mathcal{N}, \mathcal{P}, w_p, d_i, s_a, T$.

Output: Optimal solution of (TDM), if the never-meet property holds.

Step 0.

1. Calculate the time-minimal solution $y(\mathcal{A}_{\text{change}})$ if all connections are maintained by Algorithm 1.
2. (Initializations) $\text{SP}(a_0) = \emptyset, f(a_0) = 0, \text{Decompose} = \emptyset, \text{Compose} = \emptyset$.
3. (Calculate $\text{SP}(a_0)$) For all $i \in \mathcal{E}_{\text{del}}$:
 - (a) $f(a_0) = f(a_0) + G(i, d_i)$
 - (b) For all $a = (j_1, j_2) \in \mathcal{A}_{\text{change}}$ with $j_1 \in \mathcal{G}(i)$: If $y_{j_1} > 0$ then $\text{SP}(a_0) = \text{SP}(a_0) \cup \{a\}$, and $\text{Decompose} = \text{Decompose} \cup \{a\}$
4. (Optimality test) If $\text{SP}(a_0) = \emptyset$ stop: f is the optimal objective value, $\bar{z}_a = 0$ for all $a \in \mathcal{A}_{\text{change}}$

Step 1. While $\text{Decompose} \neq \emptyset$

1. Choose $a = (i_1, i_2) \in \text{Decompose}$
2. $\text{SP}(a) = \emptyset, \text{miss}(a) = w_a T, \text{maintain}(a) = G(i_2, y_{i_2})$
3. (Calculate $\text{SP}(a)$) For all $a' = (j_1, j_2) \in \mathcal{A}_{\text{change}}$ with $j_1 \in \mathcal{G}(i_2)$: If $y_{j_1} > 0$ then $\text{SP}(a) = \text{SP}(a) \cup \{a'\}$, $\text{Decompose} = \text{Decompose} \cup \{a'\}$
4. (Update Compose) If $\text{SP}(a) = \emptyset$ then $\text{Compose} = \text{Compose} \cup \{a\}$.
5. (Update Decompose) $\text{Decompose} = \text{Decompose} \setminus \{a\}$.

Step 2. While $\text{Compose} \neq \emptyset$.

1. **Choose** $a \in \text{Compose}$ with $a \in \text{SP}(\tilde{a})$
2. (**Solve subproblem** P_a) $f(a) = \min\{\text{maintain}(a), \text{miss}(a)\}$,

$$\bar{z}_a = \begin{cases} 0 & \text{if } \text{maintain}(a) \leq \text{miss}(a) \\ 1 & \text{if } \text{maintain}(a) > \text{miss}(a) \end{cases}$$

3. (**Update values for parent problem** \tilde{a})
 $\text{SP}(\tilde{a}) = \text{SP}(\tilde{a}) \setminus \{a\}$, $\text{maintain}(\tilde{a}) = \text{maintain}(\tilde{a}) + f(a)$
4. (**Update** Compose)
 $\text{Compose} = \text{Compose} \setminus \{a\}$
If $\text{SP}(\tilde{a}) = \emptyset$ **and** $\tilde{a} \neq a_0$ **then** $\text{Compose} = \text{Compose} \cup \{\tilde{a}\}$

Step 3.

1. (**Correct values for** \bar{z}_a) **For all** $a \in \mathcal{A}_{\text{change}}$: **If** $\bar{z}_a = 1$ **then set** $\bar{z}_{a'} = 0$ **for all** $a' \neq a$ **with** $a \prec a'$.
 2. **Output:** $f(a_0), \bar{z}$
-

Theorem 4 *Algorithm 2 is correct and runs in time $O(|\mathcal{A}|)$.*

Proof: We show by induction over all $a \in \mathcal{A}_{\text{change}} \cup \{a_0\}$ that $f(a)$ contains the objective value for the subproblem P_a at the end of Algorithm 2.

Start: Let $a = (i, j)$ be a maximal element of $\mathcal{A}_{\text{change}}$ (with respect to \prec). The subproblem with respect to a is (TDM-const) in the small network $\mathcal{N}(i)$, which does not contain any changing activity except of a itself, hence $\text{SP}(a) = \emptyset$ in step 2 of the algorithm. Furthermore,

$$\begin{aligned} \text{maintain}(a) &= \sum_{i' \in \mathcal{G}(i)} y_{i'} w_{i'}, \text{ and} \\ \text{miss}(a) &= Tw_a \end{aligned}$$

give the objective values of this small network when maintaining or not maintaining activity a . To see the correctness of $\text{miss}(a)$ we note that due to Lemma 3 $y_{i'} = 0$ for all $i' \in \mathcal{H}(i)$ (which equals $\mathcal{G}(i)$ in this case).

Since $a \in \text{Compose}$ we compare both values $\text{maintain}(a)$ and $\text{miss}(a)$ in step 2, and choose the better as (correct) objective value, which is then stored in $f(a)$.

Conclusion: Now take any $a = (i, j)$ and let the induction hypothesis be true for all a' with $a \prec a'$.

- If a is not maintained, we know from Lemma 3 that all connections $a' \in \mathcal{N}(j)$ are maintained and all $i' \in \mathcal{H}(j)$ satisfy $y_{i'} = 0$, i.e., the objective value is given by $\text{miss}(a)$ as calculated in step 2.
- If a is maintained the algorithm calculates in step 2 the delay which will be gained in any case, i.e., the delay of all events $i' \in \mathcal{G}(i)$ that can be reached without passing any changing activity, and store it in $\text{maintain}(a)$. All changing activities a' that can be reached from j without passing any other changing activity are stored in $\text{SP}(a)$. Due to Lemma 4 the corresponding subproblems $P_{a'}$ for $a' \in \text{SP}(a)$ are independent and have objective value $f(a')$ due to the induction hypothesis, such that $\text{maintain}(a) + \sum_{a' \in \text{SP}(a)} f(a')$ calculated in step 3 finally is the correct value of maintain .

Comparing $\text{maintain}(a)$ with $\text{miss}(a)$ and choosing the smaller of both gives the best possible choice for activity a assuming the delay y_i as given.

Finally, in step 0, the problem with the given source delays is decomposed into a set of subproblems $\text{SP}(a_0)$. All these subproblems are independent due to Lemma 4, and they are all solved optimally due to the claim above. Adding up these optimal values and adding the delay of all events which are reached before entering one of the subproblems gives the optimal objective function value $f(a_0)$.

For the time complexity we note that the number of subproblems equals the number of changing activities, which in a tree is the same as the number of events. For the decomposition steps we have to process each activity and each event exactly once, and in the composition step we need one comparison and one summation for each subproblem, and again a visit of all events. The overall time complexity is hence linear in $|\mathcal{A}|$.

QED

6 Future Research

Algorithm 2 relies on the fact that each activity $a \in \mathcal{A}_{\text{change}}$ appears in exactly one list, i.e., for each $a \in \mathcal{A}_{\text{change}}$ there exists a unique \tilde{a} such that $a \in \text{SP}(\tilde{a})$, or $a \in \text{SP}(a_0)$. If the never-meet property is not satisfied, this needs not be the case, and hence Algorithm 2 cannot be applied to (TDM) for general problems. To resolve this problem (and to obtain a heuristic by applying Algorithm 2) one can either allow that the same element is added more than once to *Compose* in

step 2 (this would mean to duplicate activities until the never-meet property is satisfied), or to update the values of *maintain* to the larger one, if an element which is already contained is added.

(TDM-const) and (TDM) can both be solved by branch and bound, taking \bar{z}_a as branching variables and reducing the number of conflicts with the never-meet property in each node. Lower bounds are derived in [Sch03]. Details and implementations are under research.

Two other directions of future research in delay management should be mentioned. First, it is a challenging task to make delay management approaches applicable in railway transportation. The drawback here is that on the tracks capacity constraints have to be taken into account. Different possibilities how such constraints can be included in the models are under research. Second, it is an open field to deal with the stochastic nature of the delays instead of assuming that the source delays are fixed.

Appendix

Proof of Theorem 1: (TDM-A) and (TDM-C) lead to the same set of optimal solutions $y \in \mathbb{R}^{|\mathcal{E}|}$.

Proof: First, using (18) the objective function of (TDM-C) can be reformulated to

$$\begin{aligned}
f_{\text{TDM-C}} &= \sum_{a=(i,j) \in \mathcal{A}^s} w_a(y_j - y_i) + \sum_{a=(i,j) \in \mathcal{A}_{\text{change}}} w_a \bar{z}_a(T - y_j) \\
&= \sum_{a=(i,j) \in \mathcal{A}^s} \sum_{p \in \mathcal{P}^s: a \in p} w_p \tilde{z}_a^p(y_j - y_i) + \sum_{a=(i,j) \in \mathcal{A}_{\text{change}}} \sum_{p \in \mathcal{P}^s: a \in p} w_p \tilde{z}_a^p \bar{z}_a(T - y_j) \\
&= \sum_{p \in \mathcal{P}^s} w_p \left(\sum_{a=(i,j) \in \mathcal{A}^s: a \in p} \tilde{z}_a^p(y_j - y_i) + \sum_{\substack{a=(i,j) \in \mathcal{A}_{\text{change}} \\ a \in p}} \tilde{z}_a^p \bar{z}_a(T - y_j) \right) \\
&=: \sum_{p \in \mathcal{P}^s} w_p C_p.
\end{aligned}$$

For the objective of (TDM-A), we define

$$A_p = y_{i(p)}(1 - z_p) + T z_p.$$

(TDM-C) \implies (TDM-A): Let $(y, \bar{z}, \tilde{z}, w)$ be feasible for (TDM-C). Define $z_p = z_p(\bar{z})$ as follows:

$$z_p(\bar{z}) = \begin{cases} 0 & \text{if } \bar{z}_a = 0 \text{ for all } a \in p \cap \mathcal{A}_{\text{change}} \\ 1 & \text{otherwise} \end{cases} \quad (35)$$

Then (6) holds due to (13), (7) holds due to (14), and (8) is trivially satisfied, if $z_p = 1$, and for $z_p = 0$ we know that $\bar{z}_a = 0$ for all $a \in p$ and hence (8) holds because of (15). This means (y, z) is feasible for (TDM-A). It remains to show that $A_p \leq C_p$. To this end, let $p = (s, i_1, \dots, i_L) \in \mathcal{P}^s$ be a path with $i(p) = i_L$.

Case 1: $\bar{z}_a = 0$ for all $a \in p \cap \mathcal{A}_{change}$. Then, we defined $z_p = 0$. From (16) we get that $\tilde{z}_a^p = 1$ for all $a \in p$. Since $y_s = 0$ we conclude that

$$C_p = \sum_{a=(i,j) \in \mathcal{A}^s: a \in p} y_j - y_i = y_{i_L} - y_s = A_p.$$

Case 2: There exists $a \in p \cap \mathcal{A}_{change}$ with $\bar{z}_a = 1$. Choose a minimal with respect to \prec with this property, say $\bar{a} = (i_{\bar{k}-1}, i_{\bar{k}})$. Then, since \bar{z}_a, \tilde{z}_a^p satisfy (16) and (17) we obtain

$$\begin{aligned} \tilde{z}_a^p &= 0 \text{ for all } a \in p \text{ with } \bar{a} \prec a \\ \tilde{z}_a^p &= 1 \text{ for all } a \in p \text{ with } a \preceq \bar{a}. \end{aligned}$$

Hence, for all $a \in \mathcal{A}_{change} \cap p$ we get

$$\tilde{z}_a^p \bar{z}_a = \begin{cases} 1 & \text{if } a = \bar{a} \\ 0 & \text{otherwise} \end{cases}$$

This yields

$$\begin{aligned} C_p &= \sum_{\substack{a=(i,j) \in \mathcal{A}^s: a \in p \\ \text{and } a \preceq \bar{a}}} y_j - y_i + (T - y_{i_{\bar{k}}}) \\ &= y_{i_{\bar{k}}} - y_s + T - y_{i_{\bar{k}}} = T = A_p, \end{aligned}$$

and consequently, $f_{\text{TDM-C}}(y, \bar{z}, \tilde{z}, w) = f_{\text{TDM-A}}(y, z(\bar{z}))$.

(TDM-A) \implies (TDM-C): Now let a feasible solution (\tilde{y}, z) of (TDM-A) be given. We replace \tilde{y} by a time-minimal solution y which satisfies $y_i \leq T$ for all $i \in \mathcal{E}$, and has equal or better objective value (see Lemma 1). Since y satisfies (6) and (7) we can construct a feasible solution for (TDM-C) according to (23), (24), and (25).

For the objective value of this solution we again compare C_p and A_p for a path $p = (s, i_1, \dots, i_L) \in \mathcal{P}^s$ and get:

Case 1: If $z_p = 0$, we get from (8) that $y_i - y_j \leq s_a$ for all $a = (i, j) \in p$. Hence, due to the definition of \bar{z}_a we conclude that $\bar{z}_a = 0$ for all $a \in p \cap \mathcal{A}_{change}$, yielding $C_p = y_{i(p)} = A_p$ analogously to Case 1 of the first part of the proof.

Case 2: Now consider the case $z_p = 1$.

Case 2a: $y_i - y_j \leq s_a$ for all $a = (i, j) \in p$, yielding that $\bar{z}_a = 0$ for all $a \in p$ and hence $C_p = y_{i(p)} \leq T = A_p$.

Case 2b: There exists $a = (i, j) \in p$ such that $y_i - y_j > s_a$. This gives us $\bar{z}_a = 1$. Choose $\bar{a} = (i_{\bar{k}-1}, i_{\bar{k}})$ minimal with respect to \prec with this property. Then, from the definition of \bar{z}_a^p we get

$$\begin{aligned}\bar{z}_a^p &= 0 \text{ for all } a \in p \text{ with } \bar{a} \prec a \\ \bar{z}_a^p &= 1 \text{ for all } a \in p \text{ with } a \preceq \bar{a}\end{aligned}$$

and analogously to Case 2 of the first part of the proof $C_p = T = A_p$.

Together, $f_{\text{TDM-A}}(\tilde{y}, z) \geq f_{\text{TDM-A}}(y, z) \geq f_{\text{TDM-C}}(y, C(y))$.

Combining both directions yields that there exists an optimal solution for (TDM-A) with delays y if and only if there exists an an optimal solution for (TDM-C) with the same delays y . QED

References

- [Ack99] T. Ackermann. *Die Bewertung der Pünktlichkeit als Qualitätsparameter im Schienenpersonennahverkehr auf Basis der direkten Nutzenmessung*. PhD thesis, Universität Stuttgart, 1999.
- [APW02] L. Anderegg, P. Penna, and P. Widmayer. Online train disposition: to wait or not to wait? *Electronic Notes in Theoretical Computer Science*, 66(6), 2002.
- [Car99] M. Carey. Ex ante heuristic measures of schedule reliability. *Transportation Research*, 53B(3):473–494, 1999.
- [EF02] O. Engelhardt-Funke. *Stochastische Modellierung und Simulation von Verspätungen in Verkehrsnetzen für die Anwendung der Fahrplanoptimierung*. PhD thesis, Univerität Clausthal, 2002.
- [EFK01] O. Engelhardt-Funke and M. Kolonko. Simulating delays for realistic timetable-optimization. In *Operations Research Proceedings 2001*, pages 9–15. Springer, 2001.
- [Elm77] S.E. Elmaghraby. *Activity Networks*. Wiley Interscience Publication, 1977.

- [GGJ⁺04] M. Gatto, B. Glaus, R. Jacob, L. Peeters, and P. Widmayer. Railway delay management: Exploring its algorithmic complexity. In *Proceedings 9th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 3111 of *LNCS*, pages 199–211, 2004.
- [GJPS04] M. Gatto, R. Jacob, L. Peeters, and A. Schöbel. The computational complexity of delay management. Technical report, ETH Zürich, Institut für Theoretische Informatik, 2004. accepted for the 31st International Workshop on Graph-Theoretic Concepts in Computer Science, 2005.
- [GJPW04] M. Gatto, R. Jacob, L. Peeters, and P. Widmayer. On-line delay management on a single train line. Technical report, ETH Zürich, 2004.
- [Gov98] R.M.P. Goverde. The max-plus algebra approach to railway timetable design. In *Computers in Railways VI: Proceedings of the 6th international conference on computer aided design, manufacture and operations in the railway and other advanced mass transit systems, Lisbon, 1998*, pages 339–350, 1998.
- [GS02] A. Ginkel and A. Schöbel. The bicriterial delay management problem. Technical report, Universität Kaiserslautern, 2002.
- [Kli00] N. Kliwer. Mathematische Optimierung zur Unterstützung kundenorientierter Disposition im Schienenverkehr. Master’s thesis, Universität Paderborn, 2000.
- [Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables*. Deutsches Zentrum für Luft- und Raumfahrt, Institut für Flugführung, Braunschweig, 1998. Habilitationsschrift.
- [NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.
- [Pee02] L. Peeters. *Cyclic Railway Timetabling Optimization*. PhD thesis, ERIM, Rotterdam School of Management, 2002.
- [RdVM98] B. De Schutter R. de Vries and B. De Moor. On max-algebraic models for transportation networks. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 457–462, Cagliari, Italy, 1998.
- [Roc84] R.T. Rockafellar. *Network Flows and Monotropic Optimization*. John Wiley, New York, 1984.

- [SBK01] L. Suhl, C. Biederbick, and N. Kliwer. Design of customer-oriented dispatching support for railways. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*, volume 505 of *Lecture Notes in Economics and Mathematical systems*, pages 365–386. Springer, 2001.
- [Sch01a] A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.
- [Sch01b] S. Scholl. Anschlussicherung bei Verspätungen im ÖPNV. Master’s thesis, Universität Kaiserslautern, 2001.
- [Sch03] A. Schöbel. Customer-oriented optimization in public transportation, 2003. Habilitationsschrift, Universität Kaiserslautern.
- [SM01] L. Suhl and T. Mellouli. Managing and preventing delays in railway traffic by simulation and optimization. In *Mathematical Methods on Optimization in Transportation Systems*, pages 3–16. Kluwer, 2001.
- [SMBG01] L. Suhl, T. Mellouli, C. Biederbick, and J. Goecke. Managing and preventing delays in railway traffic by simulation and optimization. In M. Pursula and Niittymäki, editors, *Mathematical methods on Optimization in Transportation Systems*, pages 3–16. Kluwer, 2001.
- [SvdB01] B. De Schutter and T. van den Boom. Model predictive control for railway networks. In *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 105–110, Como, Italy, 2001.
- [vE01] R.J. van Egmond. An algebraic approach for scheduling train movements. In *Proceedings of the 8th international conference on Computer-Aided Transit Scheduling, Berlin, 2000*, 2001.

Institut für Numerische und Angewandte Mathematik
Universität Göttingen
Lotzestr. 16-18
D - 37083 Göttingen

Telefon: 0551/394512

Telefax: 0551/393944

Email: trapp@math.uni-goettingen.de URL: <http://www.num.math.uni-goettingen.de>

Verzeichnis der erschienenen Preprints:

1. A. Schöbel, S. Scholl: Line Planning with Minimal Traveling Time
2. A. Schöbel Integer programming approaches for solving the delay management problem