Report in Wirtschaftsmathematik
Nr. 92/2004

Integer programming approaches for solving the delay management problem*

A. Schöbel
Technische Universität Kaiserslautern
and
Institut für Techno- und Wirtschaftsmathematik

January 2004

Abstract

In the *delay management problem* we decide how to react in case of delays in public transportation. More specific, the question is if connecting vehicles should wait for delayed feeder vehicles or if it is better to depart in time. As objective we consider the convenience over all customers, expressed as the average delay of a customer when arriving at his 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 practical data. Finally, we show how to find an optimal solution in linear time in case of the never-meet property.

1 Introduction

A major reason for complaints about public transportation is the missing punctuality, which — unfortunately — is a fact in many transportation systems. Since

*This work was supported by Stiftung Innovation Rheinland-Pfalz

it seems to be impossible to avoid delays completely, it is a necessary issue in the dispositive 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, 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 delay for passengers already within h, but also for customers who wish to get on 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.

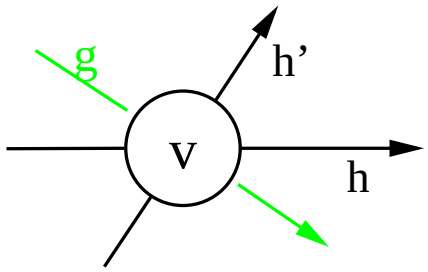

Figure 1: The wait-depart decision at one single station.

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.

The *delay management problem* is to find wait-depart decisions and a perturbed timetable in case of some known delays, not only for one single bus, but 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 he reaches his destination.
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. A recent overview is given by [Pee02]. We also refer to [Nac98, Car99, Gov98, vE01] and references therein. Note that the main difference between timetabling and delay management is that in the timetabling problem the connections are given in advance, while in the delay management problem we have 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 — be mainly tackled by simulation and expert systems. We refer to [SM97, SM99, 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 was 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].
Integer programming formulations for a simple case without slack times were developed independently in the diploma theses of [Kli00] (see also [SBK01]) and by [Sch01a]. Both assumed that the number of customers on each edge is fixed (which is not true if customers miss a connection) and hence studied an approximation of the effects of delays. An exact linear integer model for the delay management problem is presented in [Sch01b]. In all these models it is assumed that the underlying timetable is periodic with one common period T. Related work includes how to reduce delays by investing into new tracks (see [EFK01a, EFK01b, EF02]) and how to minimize the sum of waiting times of customers at their starting stations in a stochastic context (see [APW02]).

In this paper we present a new and more general integer programming formulation of the delay management problem. This new formulation contains the formulations of [Sch01a, Kli00] as special cases and is equivalent to the model in [Sch01b] in the case of one common period of time. 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 the new model (TDM-C) can be linearized if a special condition, called *the never-meet property* holds. In Section 5 we show how to solve (TDM) 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 very small event-activity network is depicted in Figure 2.

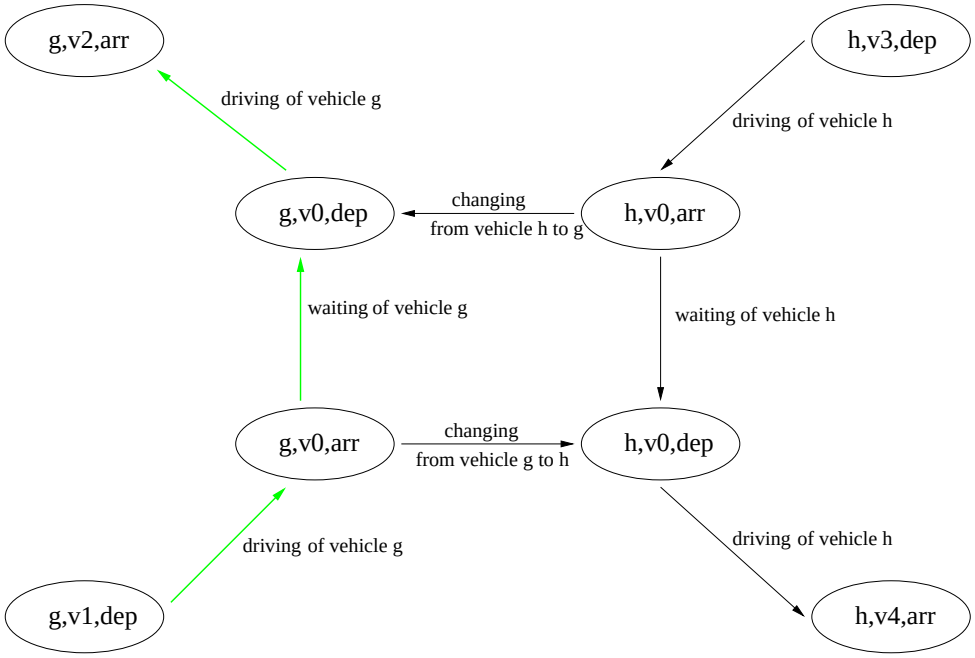

Figure 2: An event-activity network.

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) is 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 event*
- *$\mathcal{A} = \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}_{change}$ is a set of directed arcs, called* **activities**, *defined by*

$$\begin{aligned}
\mathcal{A}_{wait} &= \{((g, v, \text{arr}), (g, v, \text{dep})) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep}\} \\
\mathcal{A}_{drive} &= \{((g, v, \text{dep}), (g, u, \text{arr})) \in \mathcal{E}_{dep} \times \mathcal{E}_{arr} : \text{ vehicle } g \text{ goes} \\
&\quad \text{directly from station } v \text{ to } u\}, \\
\mathcal{A}_{change} &= \{((g, v, \text{arr}), (h, v, \text{dep})) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep} : \text{ a changing} \\
&\quad \text{possibility from vehicle } g \text{ into } h \text{ at station } v \text{ is required}\}.
\end{aligned}$$

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, a precedence relation $\prec$ between events and activities is canonically given. 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* Π is given by assigning a time Π_i to each event $i \in \mathcal{E}$ (see [Nac98]). The planned duration of activity $a = (i, j)$ is hence given by $\Pi_j - \Pi_i$. Furthermore, let L_a be the minimal duration of 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 all *source delays* are known, i.e., we have a set of (arrival) events $\mathcal{E}_{del} \subseteq \mathcal{E}_{arr}$ such that $d_i > 0$ for all $i \in \mathcal{E}_{del}$. For non-delayed events we set $d_i = 0$. If $\mathcal{E}_{del} \neq \emptyset$ some of the scheduled arrival and departure times have to be changed. The outcome is called a *perturbed timetable*.

Definition 1 *A* **perturbed timetable** x_i *for all* $i \in \mathcal{E}$ *is* **feasible**, *if*

$$x_i \geq \Pi_i + d_i \text{ for all } i \in \mathcal{E} \quad \text{and} \tag{1}$$

$$x_j - x_i \geq L_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}. \tag{2}$$

Constraint (1) requires that no event must be scheduled earlier than in the original timetable, and furthermore ensures that for all $i \in \mathcal{E}_{del}$ the source delays are taken into account. Due to constraint (2) the delay is carried over correctly from one event to the next along waiting and driving activities.

Notation 2 *A changing activity* $a = (i, j) \in \mathcal{A}_{change}$ *is called* **maintained**, *if* $x_j - x_i \geq L_a$.

In the delay management problem our goal is to identify which connections should be maintained and which can be missed. This has to be in accordance with the perturbed timetable x.

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

$$x_j - x_i \geq L_a \quad \text{for all } a = (i, j) \in \mathcal{A}^{fix}.$$

In other words, $(\mathcal{A}^{fix}, x_i)$ is feasible, if all connections $a \in \mathcal{A}^{fix}$ are maintained.

Notation 3 *The* **slack time** s_a *of activity* $a \in \mathcal{A}$ *is the time which can be saved while performing activity* a *as fast as possible, and it is given by*

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

for all three types of activities $a = (i, j) \in \mathcal{A}$.

Using Notation 3 the delays $y_i = x_i - \Pi_i$ can be used instead of the perturbed timetable. Rewriting (1) and (2) we obtain that $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}, y_i \in \mathbb{N}^{|\mathcal{E}|}$ is a feasible solution of the delay management problem, if

$$\begin{aligned} y_i &\geq d_i \quad \text{for all } i \in \mathcal{E}_{del} \\ y_i - y_j &\leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}^{fix} \end{aligned}$$

Note that a timetable would also be feasible if some vehicles depart or arrive late without any reason, i.e., without having an incoming delay. Such solutions are clearly not optimal, and hence we define a set of "most punctual" solutions.

Notation 4 *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 solution* $(\mathcal{A}^{fix}, y')$ *satisfy* $y \leq y'$. *In this case,* $(\mathcal{A}^{fix}, y)$ *is called a time-minimal solution.*

Note that given $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}$, a time-minimal solution $y = y(\mathcal{A}^{fix})$ w.r.t. $\mathcal{A}^{fix}$ can be found efficiently by one of the following methods (see also [GS02]).

Linear programming approach: We introduce variables $\bar{z}_a$ describing if connection $a \in \mathcal{A}_{change}$ is missed ($\bar{z}_a = 1$) or maintained ($\bar{z}_a = 0$). Choosing $M \geq D := \max\{d_i : i \in \mathcal{E}\}$, the following is an integer programming formulation whose solutions are exactly the time-minimal feasible solutions:

$$\begin{aligned} y_i &\geq d_i \quad \text{for all } i \in \mathcal{E}_{del} && (3) \\ y_i - y_j &\leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} && (4) \\ -M\bar{z}_a + y_i - y_j &\leq s_a \quad \text{for all } a = (i, j) \in \mathcal{A}_{change} && (5) \\ y_i &\in \mathbb{N} \quad \text{for all } i \in \mathcal{E}. \end{aligned}$$

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: The event-activity network can easily be transformed into a project network (as defined, e.g., in [Elm77]) by introducing one super-sink 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 $-s_a$ for $a \in \mathcal{A}$ and to the source delay d_i if $a = (s,i)$. Then it can be shown that the earliest possible starting time of each activity is a time-minimal solution of the delay management problem. The procedure uses the critical path method to determine the earliest starting times but is applied directly in the original network $\mathcal{N}$.

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` $\mathcal{E} = \{i_1, \ldots, i_{|\mathcal{E}|}\}$ `according to` $\prec$.

Step 2. `For` $k = 1, \ldots, |\mathcal{E}|$:

$$y_{i_k} = \max\{d_{i_k}, \max_{a=(i,i_k)\in\mathcal{A}(\mathcal{A}^{fix})} y_i - s_a\} \tag{6}$$

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.

The following result will be used throughout the paper.

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

Proof: The result can be shown easily by using 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 (6) we directly obtain that $y_j \leq D$ 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 over 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.,

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

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.
To calculate the delay of a passenger on path p we assume one (common) time period T for all vehicles, and that in the next time period all vehicles are in time. We have to distinguish the following two cases.

Case 1: If all connections on path p are 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 on 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 $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ with source delays $d_i, i \in \mathcal{E}$ and a set of weighted paths $\mathcal{P}$, find a set $\mathcal{A}^{fix} \subseteq \mathcal{A}_{change}$ and a feasible perturbed timetable $x_i = \Pi_i + y_i$, $i \in \mathcal{E}$ such that the sum over all delays over all customers is minimal.*

3 Two models for the delay management problem (TDM)

3.1 Path-based formulation

As first model we present a path-oriented description of (TDM) (based on the formulation in [Sch01b]) 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

$$\begin{aligned} y_i &\geq d_i \quad \text{for all } i \in \mathcal{E}_{del} && (7) \\ y_i - y_j &\leq s_a \quad \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} && (8) \\ -M z_p + y_i - y_j &\leq s_a \quad \text{for all } p \in \mathcal{P}, a = (i,j) \in p \cap \mathcal{A}_{change} && (9) \\ y_i &\in \mathbb{N} \quad \text{for all } i \in \mathcal{E} && (10) \\ z_p &\in \{0,1\} \quad \text{for all } p \in \mathcal{P} && (11) \end{aligned}$$

The first two constraints (7) and (8) are the same as (3) and (4). Constraint (9) corresponds to (5) but is rewritten for paths p. 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 (and weakened) by substituting the quadratic term $y_{i(p)}(1 - z_p)$ by a new variable q_p, leading to the following model (TDM-B).

(TDM-B)

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

such that (7) – (11) hold, and such that

$$\begin{aligned} -M z_p + y_{i(p)} - q_p &\leq 0 \quad \text{for all } p \in \mathcal{P} && (12) \\ q_p &\geq 0 \quad \text{for all } p \in \mathcal{P} && (13) \end{aligned}$$

Lemma 2 *The linearization is correct.*

Proof:

(TDM-A) $\Longrightarrow$ (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

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

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

(TDM-B) $\Longrightarrow$ (TDM-A): Let (y, z, q) be a feasible solution of (TDM-B). Then (y, z) is also feasible for (TDM-A). From (12) and (13) we conclude that

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

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

QED

Since the number of variables is very large in the path-oriented formulations we derive a (stronger) activity-based model for (TDM) in the next section.

3.2 Activity-based formulation

We now 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 ($\bar{z}_a = 0$). Our goal is to calculate the total delay by summing up the *additional delays* over all activities $a \in \mathcal{A}$.
To this end, 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

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

The additional event s represents the arrival of the customers at their first station (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).

$$\tilde{z}_a^p = \begin{cases} 1 & \text{if activity } a \text{ is reached on path } p \text{ without any missed} \\ & \text{connection before} \\ 0 & \text{otherwise} \end{cases}$$

$$w_a = \text{number of customers who } \textit{really} \text{ use activity } a$$

It is important to understand 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 arcs $a \in p$ or not.

(TDM-C)

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

such that

$$\begin{aligned}
y_i &\geq d_i \quad \text{for all } i \in \mathcal{E}_{del} && (14)\\
y_i - y_j &\leq s_a \quad \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} && (15)\\
-M\bar{z}_a + y_i - y_j &\leq s_a \quad \text{for all } a = (i,j) \in \mathcal{A}_{change} && (16)\\
\tilde{z}_a^p + \sum_{\substack{\tilde{a}\in p\cap\mathcal{A}_{change}:\\ \tilde{a}\prec a}} \bar{z}_{\tilde{a}} &\geq 1 \quad \text{for all } p \in \mathcal{P}^s \text{ and } a \in p && (17)\\
\tilde{z}_a^p + \bar{z}_{\tilde{a}} &\leq 1 \quad \text{for all } p \in \mathcal{P}^s \text{ and for all } a, \tilde{a} \in p \\
&\qquad \text{with } \tilde{a} \in \mathcal{A}_{change} \text{ and } \tilde{a} \prec a && (18)\\
w_a &= \sum_{p\in\mathcal{P}^s: a\in p} w_p \tilde{z}_a^p \quad \text{for all } a \in \mathcal{A}^s && (19)\\
y_i &\in \mathbb{N} \quad \text{for all } i \in \mathcal{E} \\
\bar{z}_a &\in \{0,1\} \quad \text{for all } a \in \mathcal{A}^s \\
\tilde{z}_a^p &\in \{0,1\} \quad \text{for all } p \in \mathcal{P}^s, a \in \mathcal{A}^s \\
w_a &\in \mathbb{N} \quad \text{for all } a \in \mathcal{A}^s
\end{aligned}$$

In the objective function the additional amount of delay on each activity is multiplied by the number of customers *really* using it. Restrictions (14) – (16) are the same as (3) – (5). Restriction (17) defines the values of $\tilde{z}_a^p$ in such a way, that they are forced to be 1, if no connection on path p before a has been missed, and (18) makes sure that $\tilde{z}_a^p = 0$ for all activities a after a missed connection $\tilde{a}$ on path p. Finally, (19) determines the number of customers really using activity a.

Given a feasible solution $y_i, i \in \mathcal{E}$, it can easily be extended to a feasible time-minimal solution $(y, C(y)) := (y, \bar{z}(y), \tilde{z}(\bar{z}), w(\tilde{z}))$ by

$$\bar{z}_a(y) = \begin{cases} 0 & \text{if } y_i - y_j \le s_a \\ 1 & \text{otherwise} \end{cases} \quad \text{for all } a = (i,j) \in \mathcal{A}_{change}, \tag{20}$$

$$\tilde{z}_a^p(\bar{z}) = \max\left\{ 1 - \sum_{\substack{a \in p \cap \mathcal{A}_{change}: \\ \tilde{a} \prec a}} \bar{z}_{\tilde{a}}, 0 \right\} \quad \text{for all } p \in \mathcal{P}^s, a \in p, \tag{21}$$

$$w_a(\tilde{z}) = \sum_{p \in \mathcal{P}^s : a \in p} w_p \tilde{z}_a^p \quad \text{for all } a \in \mathcal{A}^s. \tag{22}$$

The concept of $C(y)$ is important since replacing a feasible solution $(y, \bar{z}, \tilde{z}, w)$ by $(y, C(y))$ will always yield the same or a better objective function value for (TDM-C).

3.3 Relation between the models

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

Proof:
First, if

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

the objective function of (TDM-C) can be reformulated to

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

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

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

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

$$z_p(\bar{z}) = \begin{cases} 0 & \text{if } \bar{z}_a = 0 \text{ for all } a \in p \cap \mathcal{A}_{change} \\ 1 & \text{otherwise} \end{cases} \tag{23}$$

Then (7) holds due to (14), (8) holds due to (15), and (9) 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 (9) holds because of (16). This means (y, z) is feasible for (TDM-A). It remains to show that $A_p \leq C_p$. To this end, let $p = (s, i_1, \ldots, i_L) \in \mathcal{P}^s$ 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 (17) 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 (17) and (18) 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

$$\begin{aligned} C_p &= \sum_{\substack{a=(i,j) \in \mathcal{A}^s : a \in p \\ \text{and } a \preceq \bar{a}}} y_j - y_i + (T - y_{i_{\bar{k}}}) \\ &= y_{i_{\bar{k}}} - y_{i_0} + T - y_{i_{\bar{k}}} = T = A_p, \end{aligned}$$

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

(TDM-A) $\Longrightarrow$ (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 (7) and (8) we can construct a feasible solution for (TDM-C) according to (20),(21), and (22).

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

Case 1: If $z_p = 0$, we get from (9) 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 $\tilde{z}^p_a$ we get

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

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 perturbed timetable y if and only if there exists an an optimal solution for (TDM-C) with the same perturbed timetable y. QED

On a first glance, (TDM-C) does not seem to be useful for solving the delay management problem better than (TDM-A), since it is much larger:

- (TDM-A) can be linearized (see Lemma 2) while (TDM-C) is cubic.
- The number of variables in (TDM-A) is $O(|\mathcal{P}| + |\mathcal{E}|)$, but $O((|\mathcal{P}||\mathcal{A}| + |\mathcal{E}| + |\mathcal{A}_{change}|)$ in (TDM-C).

But note that (TDM-C) is more general since it allows to replace the common time period T by time periods T_a for each changing activity $a \in \mathcal{A}_{change}$. Even with a common time period T we will need (TDM-C) to solve a special case of (TDM) very efficiently in Section 5.
We remark that (TDM-C) can be used to derive the following reduction result for (TDM). Assume that the slack times are so large that the delay disappears after a few activities. Then we need not consider events which can not gain any delay in the worst-case time-minimal solution.

Lemma 3 *Let $y = y(\mathcal{A}^{change})$ be a time-minimal solution w.r.t. $\mathcal{A}^{change}$. Then there exists an optimal solution $(y^*, \bar{z}^*, \tilde{z}^*, w^*)$ of (TDM-C) such that*

- *For all $i \in \mathcal{E}$: If $y_i = 0$ then $y^*_i = 0$.*
- *For all $a = (i,j) \in \mathcal{A}_{change}$: If $y_i = 0$ then $\bar{z}^*_a = 0$.*

This kind of reduction will be referred to as *late reduction*. In the following we always can assume that

$$\mathcal{E} = \mathcal{E}_{red-late} = \{i \in \mathcal{E} : y_i(\mathcal{A}_{change}) > 0\}$$

(Note that in real-world instances, late reduction often leads to significantly smaller networks.)

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 (17), (18), and (19) in (TDM-C), and fixing

$$w_a = \sum_{p \in \mathcal{P}^s : a \in p} w_p \quad \text{for all } a \in \mathcal{A}^s \tag{24}$$

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

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

such that (14),(15),(16),(20), and (20) hold.

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

$$w_i \;=\; \sum_{p \in \mathcal{P} : i(p) = i} w_p \tag{25}$$

be the number of customers with final destination i. Since

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

we rewrite

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

Unfortunately, in general, we make a mistake by fixing the weights as above. This is illustrated in Figure 3.

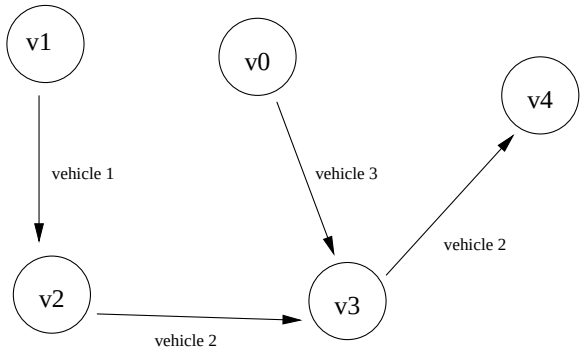

Figure 3: An example in which (TDM-const) is not correct.

We assume that 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. In this original network we consider a path $p = (v_1, v_2, v_3, v_4)$. Customers on this path use vehicle 1 until they reach station v_2; here they wish to change to vehicle 2 and to go on to stations v_3 and v_4. Suppose that vehicle 2 is not waiting for vehicle 1 at station v_2, such that the path p is missed. 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: 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 might influence decisions in the wrong way.

Fortunately, we are able to identify problem instances for which the model with constant weights is correct.

- One trivial case is given if no path in $\mathcal{P}$ contains a changing activity, i.e., no customer plans to change.
- It can be shown that the model is still correct, if we only allow paths without changing activities or of the form

  $$p = (i_1, i_2, \ldots, i_{L-2}, i_{L-1}, i_L),$$

  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 in the following.

Since $f_{\text{TDM}-\text{const}'}$ still is no linear function we first 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).

(TDM-const)

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

such that

$$\begin{aligned} y_i &\geq d_i \quad \text{for all } i \in \mathcal{E}_{del} && (26)\\ y_i - y_j &\leq s_a \quad \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} && (27)\\ -M\bar{z}_a + y_i - y_j &\leq s_a \quad \text{for all } a = (i,j) \in \mathcal{A}_{change} && (28)\\ y_i &\in \mathbb{N} \quad \forall i \in \mathcal{E}\\ \bar{z}_a &\in \{0,1\} \quad \text{for all } a \in \mathcal{A}_{change} \end{aligned}$$

It can be shown that 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 examples. To this end, we need some technical details.

Notation 5 $\mathcal{H}(i) = \{j \in \mathcal{E} :$ *there exists a (directed) path from* i *to* $j\}$

For all $j \in \mathcal{H}(i)$ we hence have $i \preceq j$.

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

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

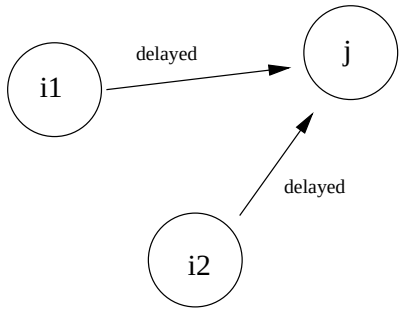

Figure 4: A case in which the never-meet-property does **not** hold.

The interpretation of the never-meet-property is the following: By calculating the time-minimal solution with respect to some given $\bar{z}$, 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 will meet, see Figure 4. Note that the formulation includes that source delays can only occur after non-delayed events.

The following property is important for proving the next theorem.

Lemma 4 *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 $\bar{z}_{\tilde{a}} = 1$ we have the following.*

1. *$y_i = 0$ for all $i \in \mathcal{H}(\tilde{j})$,*
2. *$\bar{z}_a = 0$ for all $a = (i, j)$ with $i \in \mathcal{H}(\tilde{j})$.*

Proof: Let $\mathcal{H} = \bigcup_{j \in \mathcal{E}_{del}} \mathcal{H}(j)$. Then $y_i = 0$ for all $i \notin \mathcal{H}$. Due to $\bar{z}_{\tilde{a}} = 1$ we conclude from (20) that $y_{\tilde{i}} \geq 0$ and hence $\tilde{i} \in \mathcal{H}(i^0)$ for some $i^0 \in \mathcal{E}_{del}$. Consequently, also $\tilde{j} \in \mathcal{H}(i^0)$ and due to the never-meet property, $d_{\tilde{j}} = 0$ and $i \notin \mathcal{H}$ for all $i \neq \tilde{i}$ with $(i, \tilde{j}) \in \mathcal{A}$.
We now can start induction with $\tilde{j}$ as minimal event in $\mathcal{H}(\tilde{j})$ and use (6) and the never-meet property to obtain

$$\begin{aligned} y_{\tilde{j}} &= \max_{(i,\tilde{j}) \in \mathcal{A}(\mathcal{A}^{fix})} y_i - s_a \\ &= \max_{(i,\tilde{j}) \in \mathcal{A}(\mathcal{A}^{fix}): i \notin \mathcal{H}} y_i - s_a \leq 0. \end{aligned}$$

For all other events $i \in \mathcal{H}(\tilde{j})$ we know that $i \in \mathcal{H}(i^0)$ and hence no predecessor of i can be in $\mathcal{H} \setminus \mathcal{H}(i^0)$. Note that all $j \notin \mathcal{H}$ satisfy $y_j = 0$. Furthermore, all $j \in \mathcal{H}$ satisfy $j \in \mathcal{H}(i^0)$ and hence we have for all $j \prec i$ that $y_j = 0$ due to the induction assumption. Together, we conclude that $y_i = 0$.
Finally, consider $a = (i, j)$ with $i, j \in \mathcal{H}(\tilde{j})$. Since we already have shown that $y_i = y_j = 0$ (20) 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.
First, 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.
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 $(y, C(y)) = (y, \bar{z}, \tilde{z}, w^c)$ denote a (maybe better) feasible solution. 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 $\bar{a} = (\bar{i}, \bar{j}) \in \mathcal{A}$

$$w_{\bar{a}} \neq w^c_{\bar{a}}.$$

We have to show that in this case

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

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

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

meaning that the error we make by calculating the weights will not influence the value of the objective function. Note that this is satisfied if $\bar{i} \notin \mathcal{E}_{red-late}$, since we consider a time-minimal solution.
From $w_{\bar{a}} \neq w^c_{\bar{a}}$ we get (by comparing (22) and (24), respectively), that

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

Hence there exists some path $p \in \mathcal{P}$ containing $\bar{a}$ such that $\tilde{z}^p_{\bar{a}} = 0$. Due to (21) there exists $\tilde{a} \in p$ with $\tilde{a} \prec \bar{a}$ and $\bar{z}_{\tilde{a}} = 1$. Without loss of generality let us take $\tilde{a} = (\tilde{\bar{i}}, \tilde{\bar{j}})$ 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 5.

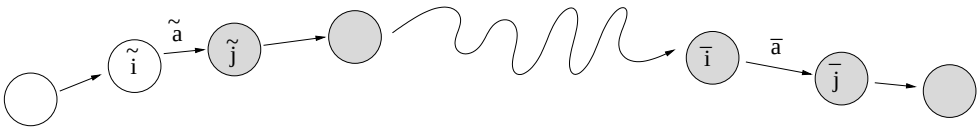

Figure 5: The path p in the proof of Theorem 2. The grey events belong to $\mathcal{H}(\tilde{j})$.

Since $\bar{i}, \bar{j} \in \mathcal{H}(\tilde{j})$ we derive from Lemma 4 that

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

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

QED

Note that the never-meet property can be tested efficiently by the forward phase of the CPM-method (with zero slack times and $\mathcal{A}^{fix} = \mathcal{A}_{change}$). This means

we have shown that (TDM-const) is correct, whenever the never-meet-property holds, and that this property can be tested efficiently. Fortunately, our numerical results indicate, that the never-meet-property often is **almost** satisfied in practice.

As test data we used a part of the public transportation network of Rheinland-Pfalz, Germany. The data consists of 823 stations, 2118 edges, and 1314 trips. As connections we use four different sets $\mathcal{U}_5, \mathcal{U}_{10}, \mathcal{U}_{30}$, and $\mathcal{U}_{60}$, 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 changing 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 nodes (events). The number of edges (activities) depends on the set U_x used and varies between 51937 and and 126122.
Our analysis with randomly chosen source delays is shown in Figures 6 and 7. In both figures the graphed 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 6 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 7 the number of conflicts with the never-meet-property is depicted as a function of the number of delayed vehicles. For this figure we assume 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 only events that can gain a delay need to be considered (see Lemma 3). Furthermore, most conflicts with the never-meet-property arise at events within the city traffic included in our data, 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 solving it. In the next section we will hence discuss how to solve (TDM-const) if the never-meet property holds.

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

5.1 The special case with zero slack times

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

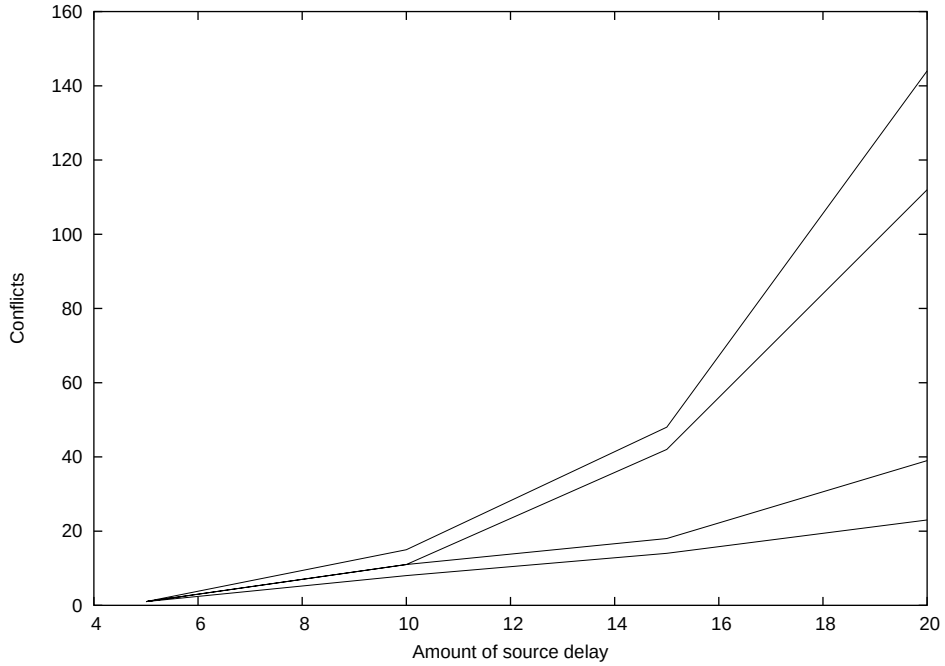

Figure 6: Conflicts with the never-meet-property as function of the source delay, if 10 vehicles are delayed.

- 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 $f_{\text{TDM}-\text{const}'}$ introduced on page 15 — simplifies to the following **linear** program. Recall that $\mathcal{E}_{del} = \{i \in \mathcal{E} : d_i > 0\}$ is the set of events with a source delay $d_i > 0$.

(TDM-const-zero)

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

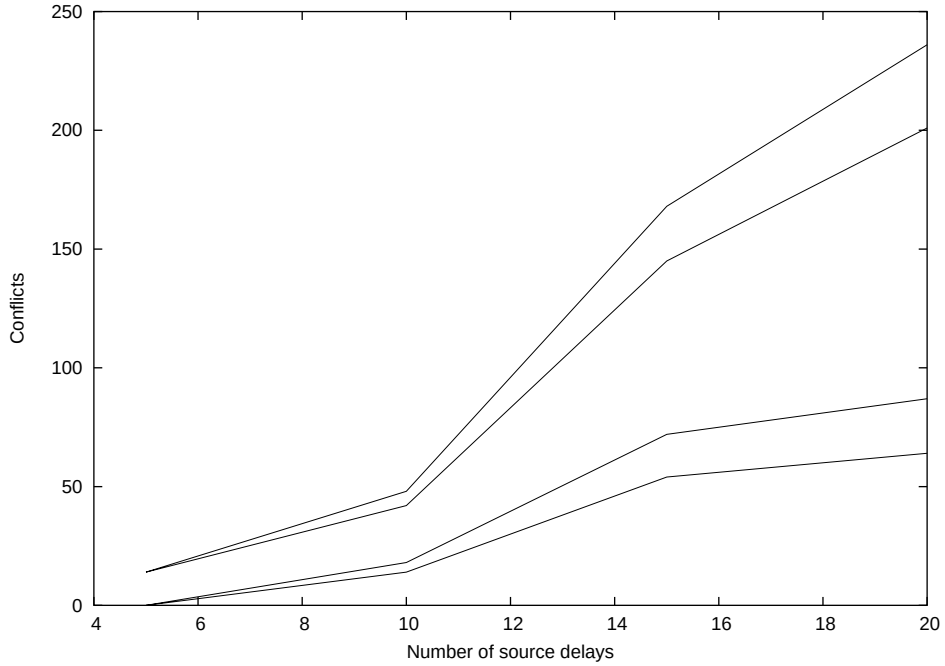

Figure 7: Conflicts with the never-meet-property as function of the number of delayed vehicles, assuming a source delay of 15 minutes for each vehicle.

such that

$$\begin{aligned} -y_i &\leq -1 \quad \text{for all } i \in \mathcal{E}_{del} && (29)\\ y_i - y_j &\leq 0 \quad \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} && (30)\\ \bar{z}_a + y_i - y_j &\leq 0 \quad \text{for all } a = (i,j) \in \mathcal{A}_{change} && (31)\\ y_i &\in \{0,1\} \quad \forall i \in \mathcal{E} \\ \bar{z}_a &\in \{0,1\} \quad \forall a \in \mathcal{A}_{change}, \end{aligned}$$

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., (24)).

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. Adding a unit matrix on the right hand side or above a totally unimodular matrix still yields a totally unimodular matrix, i.e., Φ is totally unimodular. This means, that all basic solutions of (TDM-const-zero) are integer and 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 Kliewer [Kli00] and Scholl [Sch01a], 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. Recall Notation 5 on page 17.

- First, if we fix $\bar{z}_a = 1$ for some $a = (i, j)$, this means that we can set $y_{i'} = 0$ for all $i' \in \mathcal{H}(j)$ and that all subsequent connections can be maintained (Lemma 4).
- Secondly, the problem can be decomposed into at most $|\mathcal{A}_{change}|$ independent subproblems due to the following lemma.

Lemma 5 *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.*

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

Proof: The result follows directly from the never-meet-property. QED

Suppose that some vehicle g has a delay at its arrival at station k. Then, independently of what we decide for later connections from this vehicle g to other vehicles, we can be sure that the vehicle will transfer its delay to subsequent stations, until it has been compensated by the slack times. This delay that will always be contributing to the objective is the following.

Notation 6 *Let y be a time-minimal solution of (TDM) and $i \in \mathcal{E}$ with $y_i > 0$. Then denote*

$$\begin{aligned} \mathcal{H}^{\emptyset}(i) &= \{j \in \mathcal{E} : \text{ there exists a path from } i \text{ to } j \text{ with arcs in } \mathcal{A}_{wait} \cup \mathcal{A}_{drive}\} \\ F^{\emptyset}(i, y_i) &= \sum_{j \in \mathcal{H}^{\emptyset}(i)} w_j y_j. \end{aligned}$$

Before we formally state the algorithm, consider the following example, depicted in Figure 8.

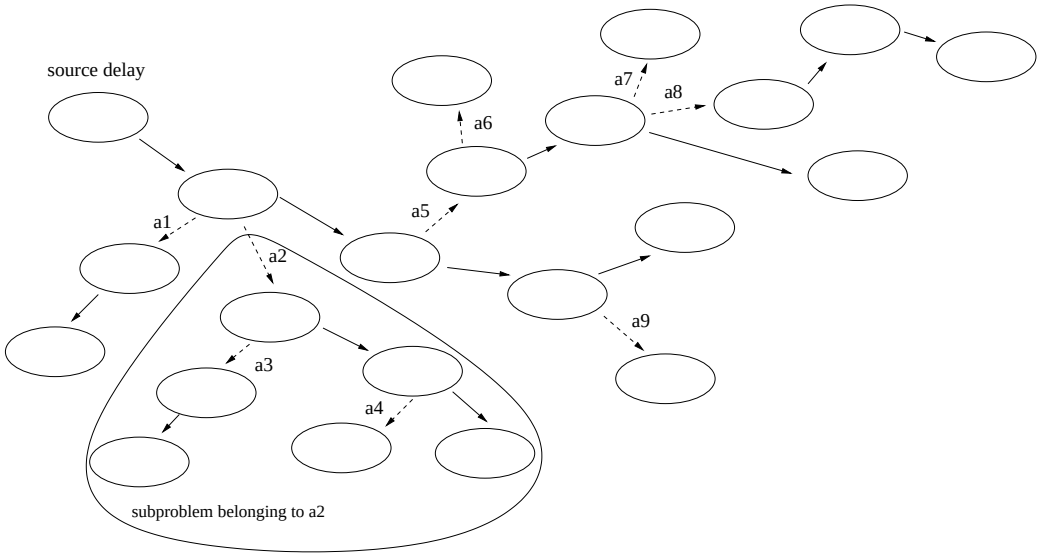

Figure 8: Decomposition of $\mathcal{N}$ in case of the never-meet-property. The changing activities are dashed.

The algorithm will first decompose P into subproblems which are collected in List(0). Each subproblem P_a is identified by a changing activity a. We obtain

$$\text{List}(0) = \{a_1, a_2, a_5, a_9\}.$$

The subproblem belonging to a_2 is depicted in Figure 8. To further decompose a subproblem P_a we store the subproblems belonging to its decomposition in List(a). In the example this gives the following lists:

$$\begin{aligned} \text{List}(a_2) &= \{a_3, a_4\} \\ \text{List}(a_5) &= \{a_6, a_7, a_8\}, \end{aligned}$$

and the lists for all other $a \in \mathcal{A}_{change}$ are empty. All subproblems that might further be decomposed are stored in Decompose, and if a subproblem cannot

further be decomposed it is collected in Compose. Hence, at the end of the decomposition step, we have

$$\text{Compose} = \{a_1, a_3, a_4, a_6, a_7, a_8, a_9\}.$$

Moreover, for each subproblem identified by changing activity a,

- maintain(a) contains the value of the objective function of the subproblem if a is maintained, and
- miss(a) contains the objective value if a is missed.

Algorithm: 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. Decompose $= \emptyset$, Compose $= \emptyset$.
2. List$(0) = \emptyset$, $f(0) = 0$
3. `For all` $i \in \mathcal{E}_{del}$:
   (a) `Calculate` y_j `for all` $j \in \mathcal{H}^{\emptyset}(i)$ `(by Algorithm 1).`
   (b) $f(0) = f(0) + F^{\emptyset}(i, d_i)$
   (c) `For all` $a = (j_1, j_2) \in \mathcal{A}_{change}$ `with` $j_1 \in \mathcal{H}^{\emptyset}(i)$: `If` $y_{j_1} > 0$
      i. List$(0) =$ List$(0) \cup \{a\}$
      ii. Decompose $=$ Decompose $\cup \{a\}$
4. `If` List$(0) = \emptyset$ `stop:` f `is the optimal objective value,` $\bar{z}_a = 0$ `for all` $a \in \mathcal{A}_{change}$.

Step 1. `While` Decompose $\neq \emptyset$

1. `Choose` $a \in$ Decompose, $\tilde{a} \in \mathcal{A}_{change}$ `with` $a = (i_1, i_2) \in$ List$(\tilde{a})$
2.

$$\begin{aligned} \text{List}(a) &= \emptyset, \\ \text{maintain}(a) &= 0, \\ \text{miss}(a) &= w_a T. \end{aligned}$$

3. $y_{i_2} = \max\{y_{i_1} - s_a, 0\}$
4. `Calculate` $\mathcal{H}^{\emptyset}(i_2)$
5. `Calculate` y_j `for all` $j \in \mathcal{H}^{\emptyset}(i_2)$ `(by Algorithm 1)`

6. $\text{maintain}(a) = \text{maintain}(a) + F^{\emptyset}(i_2, y_{i_2})$

7. `For all` $a' = (j_1, j_2) \in \mathcal{A}_{change}$ `with` $j_1 \in \mathcal{H}^{\emptyset}(i_2)$: `If` $y_{j_1} > 0$

   (a) $\text{List}(a) = \text{List}(a) \cup \{a'\}$

   (b) $\text{Decompose} = \text{Decompose} \cup \{a'\}$

8. `If` $\text{List}(a) = \emptyset$ `then` $\text{Compose} = \text{Compose} \cup \{a\}$.

9. $\text{Decompose} = \text{Decompose} \setminus \{a\}$.

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

1. `Choose` $a \in \text{Compose}$, $\tilde{a} \in \mathcal{A}_{change}$ `with` $a \in \text{List}(\tilde{a})$

2. `Define`

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

3.

$$\begin{aligned} \text{List}(\tilde{a}) &= \text{List}(\tilde{a}) \setminus \{a\} \\ \text{maintain}(\tilde{a}) &= \text{maintain}(\tilde{a}) + f(a) \\ \text{Compose} &= \text{Compose} \setminus \{a\} \end{aligned}$$

4. `If` $\text{List}(\tilde{a}) = \emptyset$ `then` $\text{Compose} = \text{Compose} \cup \{\tilde{a}\}$

Step 3. `Output:` $f(0), \bar{z}$

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

Proof: We show by induction that at the end of Algorithm 2 $f(a)$ contains the objective value for the subproblem P_a. P_a is defined as (TDM-const) in the following network determined by $a = (i, j)$ and some delay y_i:

$$\mathcal{N}_a = (\mathcal{H}(i), \mathcal{A}(\mathcal{H}(i)),$$

where

$$\mathcal{A}(\mathcal{H}(i)) = \{(j_1, j_2) \in \mathcal{A} : j_1, j_2 \in \mathcal{H}(i)).$$

The network belonging to P_{a_2} in the example is depicted in Figure 8.

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

$$\begin{aligned} \text{maintain}(a) &= \sum_{i' \in \mathcal{H}(i)} y_{i'} w_{i'}, \text{ and} \\ \text{miss}(a) &= T w_a \end{aligned}$$

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 4 $y_{i'} = 0$ for all $i' \in \mathcal{H}(i)$.

Since $a \in$ Compose we compare both values $\text{maintain}(a)$ and $\text{miss}(a)$ in step 3, 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 $a \prec a'$. Then, if a is not maintained, we know from Lemma 4 that all connections $a' \in \mathcal{H}^{\emptyset}(i)$ are maintained and all $i' \in \mathcal{H}^{\emptyset}(i)$ satisfy $y_{i'} = 0$, i.e., the objective value is in this case given by $\text{miss}(a)$ as calculated in step 2.

For maintaining activity a the algorithm calculates in step 2 the delay which will be gained for sure, i.e., the delay of all events $i' \in \mathcal{H}^{\emptyset}(i)$ that can be reached without passing any changing activity, and stores it in $\text{maintain}(a)$. All changing activities a' that can be reached from j without passing any other changing activity are stored in $\text{List}(a)$. Each of these activities a' forms an independent subproblem on the smaller network $\mathcal{N}_{a'}$, since for $a_1 = (i_1, j_1), a_2 = (i_2, j_2) \in \text{List}(a)$ we have that

$$\mathcal{H}(i_1) \cap \mathcal{H}(i_2) = \emptyset$$

according to Lemma 5.

In step 3, we add up

$$\text{maintain}(a) + \sum_{a' \in \text{List}(a)} f(a').$$

This sum contains the best possible objective value for (TDM-const) on $\mathcal{N}_a$, if a is maintained, since $f(a')$ contains the optimal objective value of subproblem (TDM-const) on $\mathcal{N}_{a'}$ due to the induction hypothesis.

Again, comparing the above sum (stored in $\text{maintain}(a)$) 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, given in List(0). All these subproblems are independent due to Lemma 5, 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 $f(0)$.

For the time complexity we see that the number of subproblems equals the number of changing activities. For the decomposition step we have to process each activity exactly once, and in the composition step we need one comparison and one summation for each subproblem. The overall time complexity is hence linear in $|\mathcal{A}|$.

QED

6 Conclusions

The algorithm relies on the fact that each activity $a \in \mathcal{A}_{change}$ appears in exactly one list, i.e., for each $a \in \mathcal{A}_{change}$ there exists a unique $\tilde{a}$ such that $a \in \text{List}(\tilde{a})$, or $a \in \text{List}(0)$. 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, 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.

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.

[EFK01a] O. Engelhardt-Funke and M. Kolonko. Cost-benefit analysis of investments into railway networks with randomly pertubed operations. In S. Voß and J. Daduna, editors, *Computer-Aided Transit Scheduling*, volume 505 of *Lecture Notes in Economics and Mathematical systems*, pages 442–459. Springer, 2001.

[EFK01b] 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.

[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. Kliewer. 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. Kliewer. 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] S. Scholl. Anschlusssicherung bei Verspätungen im ÖPNV. Master's thesis, Universität Kaiserslautern, 2001.

[Sch01b] A. Schöbel. A model for the delay management problem based on mixed-integer programming. *Electronic Notes in Theoretical Computer Science*, 50(1), 2001.

[Sch03] A. Schöbel. Customer-oriented optimization in public transportation, 2003. Habilitationsschrift, Universität Kaiserslautern.

[SM97] L. Suhl and T. Mellouli. Supporting planning and operation time control in transportation systems. In *Operations Research Proceedings 1996*, pages 374–379. Springer, 1997.

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

[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.