To wait or not to wait? The bicriteria delay management problem in public transportation

Andreas Ginkel and Anita Schöbel*
Georg-August Universität Göttingen

May 9, 2007

Abstract

Assume that a train reaches a station with delay. At the station there is a bus ready to depart. The question if such a bus should wait for the delayed train or if it should depart on time is called the *delay management problem*. Different single objective functions for this problem have been introduced and analyzed. In this paper, we present a *bicriteria* model for the delay management problem, taking into account both the delay of the vehicles and the number of passengers who miss a connection. Our model does not depend on detailed data about the passengers and can hence easily be implemented in practice.

To analyze the problem, we present an integer programming formulation and a graph-theoretic approach which is based on discrete time/cost trade-off project networks. Using results of project planning we develop an efficient solution method. We tested our procedure using real-world data. The results show the applicability of the approach.

1 Introduction and Literature Review

Since delays are a major reason for complaints about public transportation, many railway companies increase their efforts in avoiding delays. This can be done by determining delay-resistant timetables (Liebchen and Stiller 2006; Liebchen et al. 2007) or by including this goal in earlier planning steps, e.g. during the line planning process (Schöbel and Schwarze 2006). However, many delays are not avoidable. In this paper we aim to minimize passengers' inconvenience in case of delays. To this end, let us first discuss which effect the delay of a bus or train may have on the passengers: If a vehicle reaches a station with a delay, passengers

*partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

getting out there reach their destination with this delay. The situation becomes worse if a passenger wants to change from the delayed vehicle into another bus or train, but misses the connection. In this case he may have to wait a long time before the next vehicle towards his destination arrives.

The situation for one single station, and only one wait-depart decision is depicted in Figure 1. Here, vehicle g arrives with a delay. The question is to decide if the vehicles h and h' should wait for g or depart on time.

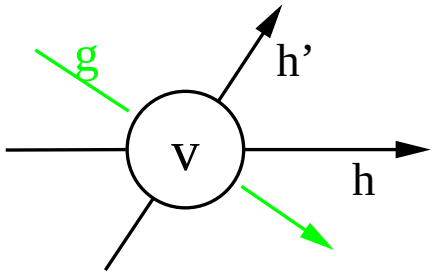

Figure 1: The wait-depart decision at one single station.

It turns out that there are two conflicting goals: If all departing vehicles wait for delayed feeder trains (or buses), no connections will be missed, but the sum of delays over all vehicles will be enormous. On the other hand, if all vehicles depart as punctual as possible, the number of delayed vehicles is minimized, but many connections will be missed. We hence face a bicriteria problem. However, only the following two (single-criterion) approaches have been investigated in the literature so far:

1. The weighted sum of both functions has been minimized by an enumeration procedure and by a greedy heuristic within a max-plus algebraic model (De Schutter 1998; De Schutter and van den Boom 2001).

2. Another possibility is to minimize the average delay of the passengers. Linear integer programming formulations for this approach can be found in (Schöbel 2001; Schöbel 2007) and were further developed in (Giovanni et al. 2006). The NP-completeness of this problem has been shown in (Gatto et al. 2005). Dynamic programming has been used in (Gatto et al. 2004) to identify polynomially solvable cases. A branch & bound approach is presented in (Schöbel 2006).

Due to the size and complexity of the problem other approaches mainly use simulation and expert systems. We refer to (Suhl et al. 2001) 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 (Ackermann 1999; Suhl and Mellouli 2001). Concerning applicability, a project in cooperation with the

largest German railway company *Deutsche Bahn* is described in (Bissantz et al. 2005).

A drawback of many of the existing models concerns their applicability for real-world problems, since in many cases very detailed data about the passengers is required: As input, an origin-destination matrix (OD-matrix) is needed, which provides information not only on the relations but also on the time at which the passengers start their journeys. While OD-matrices are sometimes known for railway transportation, many bus companies do not have this information, and time-dependent OD-matrices are usually not available at all. In particular, nearly nothing is known about passengers transferring between different transportation companies. The bicriteria approach presented in this paper does not need such detailed data and can hence be implemented in practice.

The remainder of the paper is structured as follows: We prove the NP-hardness of the *bicriteria delay management problem (BDM)* and present an integer programming formulation of (BDM) in Section 2. In Section 3 we discuss a simple variant of the problem, in which we assume that the wait-depart decisions are already given. We show the relation of this problem to project planning and how it can be solved by a shrinking method. Based on these results, we finally present an exact solution method for the bicriteria delay management problem in Section 4. Numerical results using real-world data of a large traffic association in the Palatinate region of Germany are discussed in Section 5.

2 A model for the bicriteria delay management problem

Let a public transportation network, a set of vehicles (buses or trains), and a timetable be given. Assume that some (unexpected) source delays occur. In the *bicriteria delay management problem*, the goal is to decide if vehicles should depart on time or wait for possibly delayed feeder vehicles. Our goal is to minimize the following two objective functions:

- The sum of all delays of all vehicles at all stations, and
- the weighted number of missed connections.

In order to formulate (BDM), we use the concept of *event-activity networks*, see e.g., (Nachtigall 1998) and references therein. Given a public transportation network PTN $= (V, E)$ and the set of vehicles $\mathcal{F}$, the corresponding event-activity-network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ is constructed as follows. We define a set of nodes, representing *arrival events* and *departure events*

$$\mathcal{E} := \mathcal{E}_{arr} \cup \mathcal{E}_{dep},$$

and a set of arcs, representing *driving activities*, *waiting activities* (relevant for the vehicles) and *changing activities* (relevant for the passengers)

$$\mathcal{A} := \mathcal{A}_{drive} \cup \mathcal{A}_{wait} \cup \mathcal{A}_{change}$$

as follows:

$$\begin{aligned}
\mathcal{E}_{arr} &= \{(v, g, \text{arr}) : \text{ vehicle } g \in \mathcal{F} \text{ arrives at station } v \in V\}, \\
\mathcal{E}_{dep} &= \{(v, g, \text{dep}) : \text{ vehicle } g \in \mathcal{F} \text{ departs from station } v \in V\}, \\
\mathcal{A}_{drive} &= \{((v, g, \text{dep}), (v', g, \text{arr})) \in \mathcal{E}_{dep} \times \mathcal{E}_{arr} : \text{ vehicle } g \text{ goes} \\
&\quad \text{directly from station } v \text{ to } v'\}, \\
\mathcal{A}_{wait} &= \{((v, g, \text{arr}), (v, g, \text{dep})) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep}\} \\
\mathcal{A}_{change} &= \{((v, g, \text{arr}), (v, h, \text{dep})) \in \mathcal{E}_{arr} \times \mathcal{E}_{dep} : \text{ a changing} \\
&\quad \text{possibility from vehicle } g \text{ into } h \text{ at station } v \text{ should be provided}\}.
\end{aligned}$$

If $a \in \mathcal{A}$ is an activity joining events i and j of $\mathcal{E}$, we write $a = (i, j)$. Note that an event-activity network is a time-expanded network and hence never contains directed cycles. An example for an event-activity network representing the PTN of Figure 5 is given in Figure 2.

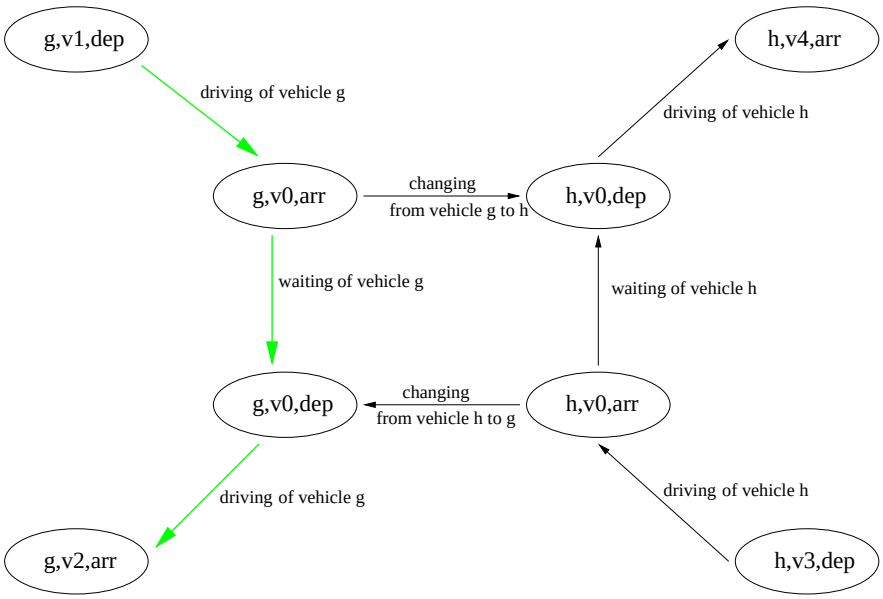

Figure 2: An event-activity network.

A *timetable* $\Pi = (\Pi_i)_{i \in \mathcal{E}}$ specifies the arrival and the departure time of each vehicle at each station, i.e. it assigns a time Π_i to each event $i \in \mathcal{E}$. In case of a

delay of event i, its actual departure time (if $i \in \mathcal{E}_{dep}$) or its actual arrival time (if $i \in \mathcal{E}_{arr}$) y_i does not coincide with the scheduled time Π_i, i.e., we have

$$y_i > \Pi_i.$$

Also the arrival and departure times of other subsequent events may have to be updated in this case. To obtain feasibility of such an updated timetable, we need some further notation. For an activity $a = (i, j) \in \mathcal{A}$ its scheduled duration is given by $\Pi_j - \Pi_i$. On the other hand, let $L_a = L_{ij}$ be the technical minimal duration which is needed to perform activity a. In case that $a \in \mathcal{A}_{drive}$ is a driving activity, L_a is the driving time needed, if the vehicle drives with maximum speed, and in case of $a \in \mathcal{A}_{change}$, L_a may refer to the time needed for the corresponding transfer. We assume that the given timetable is *feasible*, i.e.,

$$\Pi_j - \Pi_i \geq L_a$$

for all activities $a = (i, j) \in \mathcal{A}$. The *slack time* $s_a = \Pi_j - \Pi_i - L_a$ is the time which can be saved if activity $a = (i, j)$ is performed as fast as possible.

We finally assume that all *source delays* d_a are known, where a source delay of $d_a = 0$ means that the activity can be performed within the scheduled time. If some activities have a source delay $d_a > 0$, the original timetable Π may not be realizable any more. In this case, the original timetable needs to be updated. A *feasible* update is called a *disposition timetable*. It is given by updated times y_i for all events $i \in \mathcal{E}$ such that

$$\begin{aligned} y_i &\geq \Pi_i \text{ for all } i \in \mathcal{E} \text{ and }, & (1)\\ y_j - y_i &\geq L_a + d_a \text{ for all } a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}. & (2) \end{aligned}$$

Constraint (1) means that an event must not be scheduled earlier as in the original timetable, while constraint (2) ensures that the source delays are taken into account, and that a delay is transferred correctly along driving and waiting activities. The model for the bicriteria delay management problem is hence the following.

(BDM) Given $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, a feasible timetable Π_i for all $i \in \mathcal{E}$, minimal durations L_a for all activities $a \in \mathcal{A}$, the number w_a of passengers who want to transfer at activity $a \in \mathcal{A}_{change}$, and source delays d_a, determine a disposition timetable $y = (y_i)_{i \in \mathcal{E}}$ such that the following two objectives are minimized simultaneously.

The sum of all delays over all vehicles and all stations: $\sum_{i \in \mathcal{E}} y_i - \Pi_i$, which is equivalent to the minimization of

$$f_{\mathcal{E}}(y) = \sum_{i \in \mathcal{E}} y_i$$

The weighted number of missed connections given as

$$f_{\mathcal{A}}(y) = \sum_{a=(i,j)\in\mathcal{A}_{change}: y_j - y_i < L_a} w_a$$

What we mean by "minimizing simultaneously" is to find Pareto solutions which are described next: Let y_1, y_2 denote two disposition timetables. Then y_1 *dominates* y_2 if

$$\begin{aligned} f_{\mathcal{E}}(y_1) &\leq f_{\mathcal{E}}(y_2) \text{ and} \\ f_{\mathcal{A}}(y_1) &\leq f_{\mathcal{A}}(y_2), \end{aligned}$$

where at least one of the inequalities is strict. A *Pareto solution* is a disposition timetable which is not dominated by any other disposition timetable. For an introduction into multicriteria combinatorial optimization we refer e.g. to (Ehrgott 2005).

The objective space of (BDM) is given by

$$\left\{ \begin{pmatrix} f_1(y) \\ f_2(y) \end{pmatrix} : y \text{ is a disposition timetable} \right\}.$$

A point $\begin{pmatrix} f_{\mathcal{E}}(y^*) \\ f_{\mathcal{A}}(y^*) \end{pmatrix}$ in objective space is called an *efficient* point if y^* is a Pareto solution. In this paper we will present an algorithm for determining all efficient points, each of them together with a Pareto solution y^*.
Note that instead of minimizing the sum of all delays it is also possible to consider arrival delays only, or to deal with delayed events instead of delayed activities (Schöbel 2006).

Our first result clarifies the complexity status of (BDM).

Theorem 1 *(BDM) is NP-hard, even if no two connections are contained in the same connected component of the given* PTN*.*

Proof: We reduce (BDM) to the knapsack problem, which is NP-hard (Garey and Johnson 1979). Given an instance of the knapsack problem, i.e., n items, each of them with cost c_k and benefit b_k, $k = 1, \ldots, n$, and threshold parameters C and B does there exist a subset of items with total weight less than or equal to C and a total benefit of at least B?

Given an instance of the knapsack problem, we construct an instance of (BDM), in which each wait-depart decision corresponds to one of the n given items of the knapsack problem. To this end, we first construct a public transportation network PTN $= (V, E)$ as follows: V consists of $3n$ nodes, numbered by v_{1k}, v_{2k}, v_{3k}, $k = 1, \ldots, n$. The edge set is given as

$$E = \{(v_{1k}, v_{2k}), (v_{2k}, v_{3k}) : k = 1, \ldots, n\}.$$

Construct $2n$ vehicles $t_1, \ldots, t_n, \bar{t}_1, \ldots, \bar{t}_n$ where t_k goes from v_{1k} to v_{2k}, while $\bar{t}_k$ starts at v_{2k} and arrives at v_{3k}, see Figure 3. We assume the (easiest) case of zero slack times, i.e., $L_a := \Pi_j - \Pi_i$ for all $a = (i, j)$.

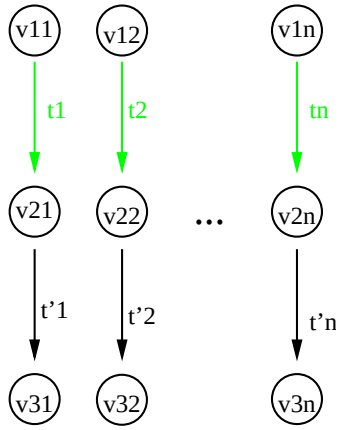

Figure 3: Reduction of (BDM) to the knapsack problem.

There are n changing activities $a_k, k = 1, \ldots, n$, where a_k refers to the changing activity at station v_{2k}, i.e., from (t_k, v_{2k}, arr) to (t'_k, v_{2k}, dep). As weight for a_k we set $w_{a_k} := b_k$.
Furthermore, assume that each of the driving activities from (t_k, v_{1k}, dep) to (t_k, v_{2k}, arr) is delayed by c_k, $k = 1, \ldots, n$, and we set $C' := \sum_{k=1}^{n} c_k + 2C$ and $B' := \sum_{k=1}^{n} b_k - B$.
Claim: There exists a solution to (BDM) with $f_{\mathcal{E}} \leq C'$ and $f_{\mathcal{A}} \leq B'$ if and only if the instance of the knapsack problem can be answered by yes.
To see this equivalence, let y be a solution of (BDM) and define

$$\mathcal{U}^* := \{a = (i, j) \in \mathcal{A}_{change} : y_j - y_i \geq L_a\}$$

as the set of maintained connections, each of them corresponding to an item packed into the rucksack. The set of vehicles which do not depart on time hence is

$$\mathcal{K}^* = \{k : a_k \in \mathcal{U}^*\} = \{k : t'_k \text{ waits for } t_k \text{ at } v_{2k}\}.$$

We obtain

$$\begin{aligned}
& f_{\mathcal{E}}(y) \leq C' \\
\iff & \sum_{k=1}^{n} y_{(t_k, v_{2k}, arr)} - \Pi_{(t_k, v_{2k}, arr)} + \sum_{k \in \mathcal{K}^*} y_{(t'_k, v_{2k}, dep)} - \Pi_{(t'_k, v_{2k}, dep)} \\
& + \sum_{k \in \mathcal{K}^*} y_{(t'_k, v_{3k}, arr)} - \Pi_{(t'_k, v_{3k}, arr)} \leq C' \\
\iff & \sum_{k=1}^{n} c_k + 2 \sum_{k \in \mathcal{K}^*} c_k \leq \sum_{k=1}^{n} c_k + 2C \\
\iff & \sum_{i \in \mathcal{U}^*} c_k \leq C
\end{aligned}$$

and

$$\begin{aligned}
& f_{\mathcal{A}}(y) \leq B' \\
\iff & \sum_{a \notin \mathcal{U}^*} w_a \leq B' \\
\iff & \sum_{k \notin \mathcal{K}^*} b_k \leq \sum_{k=1}^{n} b_k - B \\
\iff & \sum_{k \in \mathcal{U}^*} b_k \geq B.
\end{aligned}$$

This establishes the claim and consequently shows the NP-completeness.

QED

We now formulate (BDM) as a bicriteria (linear) integer program. We use two types of variables:

- The disposition timetable is given by integer variables y_i for all events $i \in \mathcal{E}$, while
- the binary variables z_a determine if a connection $a \in \mathcal{A}_{change}$ is maintained or not. More precisely,

$$z_a = \begin{cases} 1 & \text{if } a \text{ is not maintained} \\ 0 & \text{if } a \text{ is maintained.} \end{cases}$$

Furthermore, let $M > \sum_{a \in \mathcal{A}} d_a$. The integer programming formulation of (BDM) is the following.

$$\min \begin{pmatrix} \sum_{a \in \mathcal{A}} w_a z_a \\ \sum_{i \in \mathcal{E}} y_i \end{pmatrix} \tag{3}$$

such that

$$\begin{array}{rrcll} & y_i & \geq & \Pi_i & \text{for all } i \in \mathcal{E} \\ & y_j - y_i & \geq & L_a + d_a & \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \\ Mz_a + & y_j - y_i & \geq & L_a & \text{for all } a = (i,j) \in \mathcal{A}_{change} \\ & y_i & \geq & 0 & \text{for all } i \in \mathcal{E} \\ & z_a & \in & \{0,1\} & \text{for all } a \in \mathcal{A} \end{array}$$

This formulation is correct due to the following result.

Lemma 1 *For M sufficiently large we have that*

$$z_a = 1 \Longleftrightarrow y_j - y_i < L_a$$

in any Pareto optimal solution of (3).

Proof: Let a feasible solution (y,z) with objective value $\begin{pmatrix} z_{\mathcal{A}} \\ z_{\mathcal{E}} \end{pmatrix}$ be given. Take $a = (i,j) \in \mathcal{A}$. First note, that for $z_a = 0$, the constraint $Mz_a + y_j - y_i \geq L_a$ yields $y_j - y_i \geq L_a$. On the other hand, suppose that $z_a = 1$, but $y_j - y_i \geq L_a$. In this case we change z_a from 1 to 0 to obtain a new feasible solution with objective value

$$\begin{pmatrix} z_{\mathcal{A}} - 1 \\ z_{\mathcal{E}} \end{pmatrix} < \begin{pmatrix} z_{\mathcal{A}} \\ z_{\mathcal{E}} \end{pmatrix}$$

strictly dominating (y,z). Consequently, (y,z) is not Pareto optimal.

QED

3 The delay management problem with fixed connections

An important building block for the main algorithm presented in Section 4 is the solution of the *delay management problem with fixed connections (DM-Fix)*. In this variant we assume that the set of connections $\mathcal{A}_{fix} \subseteq \mathcal{A}_{change}$ which have to be maintained has been fixed beforehand. In the resulting problem we only have to calculate the disposition timetable y_i for all events $i \in \mathcal{E}$. We will use this (simple) problem to construct a project network and solve it by the *shrinking method* of project planning. The resulting procedure will be needed in the approach for solving the bicriteria delay management problem (BDM).

(DM-Fix) Given $\mathcal{N} = (\mathcal{E}, \mathcal{A})$, a feasible timetable Π_i for all $i \in \mathcal{E}$, durations L_a for all activities $a \in \mathcal{A}$, source delays d_a and a set of changing activities $\mathcal{A}_{fix} \subseteq \mathcal{A}_{change}$ determine a disposition timetable $y_i, i \in \mathcal{E}$ satisfying

$$y_j - y_i \geq L_a \text{ for all } a = (i,j) \in \mathcal{A}_{fix} \tag{4}$$

and minimizing $f_{\mathcal{E}} = \sum_{i \in \mathcal{E}} y_i$.

Note that (DM-Fix) can be solved easily by setting $z_a = 1$ for all $a \in \mathcal{A}_{fix}$ in the integer program (3) which consequently simplifies to the following one-criteria linear program.

$$\min \sum_{i \in \mathcal{E}} y_i \tag{5}$$

such that

$$\begin{array}{rcll} y_i & \geq & \Pi_i & \text{for all } i \in \mathcal{E} \\ y_j - y_i & \geq & L_a + d_a & \text{for all } a = (i,j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}_{fix} \\ y_i & \geq & 0 & \text{for all } i \in \mathcal{E}. \end{array}$$

The coefficient matrix of (5) is totally unimodular (Schöbel 2006), such that the solution of the linear program is integer. Hence the disposition timetable y can assumed to be in minutes (which is usually required in practice), and (DM-Fix) can be solved efficiently by linear programming methods. Other approaches which give more insight when solving the bicriteria problem (BDM) will be presented in the following.

We first construct a *project network* $\bar{\mathcal{N}}$ from the event-activity network $\mathcal{N}$ as follows.

$$\bar{\mathcal{E}} := \mathcal{E} \cup \{s,t\} \tag{6}$$

$$\bar{\mathcal{A}} := \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}_{fix} \cup \{(s,i) : i \in \mathcal{E}\} \cup \{(i,t) : i \in \mathcal{E}\} \tag{7}$$

$$\bar{L}_a := \begin{cases} L_a + d_a & \text{if } a \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}_{fix} \\ \Pi_i & \text{if } a = (s,i), i \in \mathcal{E} \\ 0 & \text{if } a = (i,t), i \in \mathcal{E} \end{cases} \tag{8}$$

The additional arcs $a = (s,i) \in \bar{\mathcal{A}} \setminus \mathcal{A}$ are called *timetable arcs*.

The resulting project network $\bar{\mathcal{N}} = (\bar{\mathcal{E}}, \bar{\mathcal{A}})$ is an acyclic digraph with exactly one source s and one sink t. In project planning, s represents the start of the project and t its completion. $\bar{\mathcal{A}}$ refers to the activities that have to be performed. Note that for each *event* $i \in \bar{\mathcal{E}}$ there exists an s-t-path in $\bar{\mathcal{N}}$ containing i. Furthermore, the network does not contain directed cycles, such that a partial order can be defined as follows: For $a, a' \in \bar{\mathcal{A}}$ we order $a' \prec a$ if a' occurs before a on a path from s to t.
Activity a can only be performed if all activities a' with $a' \prec a$ have been completed. The classical goal in project planning is to find the *makespan*, i.e., the minimal completion time of the whole project. It is well known that this can be done efficiently, e.g., by the critical path method (CPM) (Elmaghraby 1977).

Applying the forward phase of the critical path method (CPM) to the project network $\bar{\mathcal{N}}$ recursively yields a disposition timetable $\bar{y}$ with

- $\bar{y}_s := 0$
- $\bar{y}_j = \max_{i:(i,j)\in\bar{\mathcal{A}}}\{\bar{y}_i + \bar{L}_{ij}\}$

minimizing the completion time of the project. But in delay management we are not interested in minimizing the completion time of the project, which is the time (at the end of the planning period, e.g. at night) when the last bus reaches its depot. We are, however, interested in minimizing the *sum* $f_{\mathcal{E}}$ of all delays over all vehicles during the planning period. The next result shows that CPM achieves this goal.

Theorem 2 *The forward phase of the critical path method (CPM) finds an optimal solution of (DM-Fix).*

Proof: Let $\bar{y}$ be the timetable calculated by (CPM) and let y be another feasible timetable such that

$$f_{\mathcal{E}}(y) = \sum_{i\in\mathcal{E}} y_i < \sum_{i\in\mathcal{E}} \bar{y}_i = f_{\mathcal{E}}(\bar{y}).$$

Take j^* minimal w.r.t. $\prec$ satisfying $y_{j^*} < \bar{y}_{j^*}$. Then

$$\begin{aligned} y_{j^*} < \bar{y}_{j^*} &= \max_{i:(i,j^*)\in\bar{\mathcal{A}}} \bar{y}_i + \bar{L}_{ij^*} \\ &\leq \max_{i:(i,j^*)\in\bar{\mathcal{A}}} y_i + \bar{L}_{ij^*} \\ &= y_{i^*} + \bar{L}_{i^*j^*}, \end{aligned}$$

meaning that there exists $i^* \prec j^*$ which is not completed at time y_{j^*}.

Case 1: $i^* \in \mathcal{E}$ and $(i^*, j^*) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}$. Then $\bar{L}_{i^*j^*} = L_{i^*j^*} + d_{i^*j^*}$, but $y_{j^*} < y_{i^*} + L_{i^*j^*}$ contradicts the feasibility of y, see constraint (2).

Case 2: $i^* \in \mathcal{E}$ and $(i^*, j^*) \in \mathcal{A}_{fix}$. Then $y_{j^*} < y_{i^*} + L_{i^*j^*}$ does not maintain the connection due to the changing arc (i^*j^*) and is hence not feasible for (DM-Fix), see (4).

Case 3: $i^* = s$. In this case $y_{i^*} = 0$ and $\bar{L}_{i^*j^*} = \Pi_{j^*}$. Hence, $y_{j^*} < 0 + \Pi_{j^*}$ which contradicts (1). Consequently, y is not feasible.

QED

Unfortunately, CPM is not suitable to solve bicriteria project planning problems like (BDM). But there are other possibilities of minimizing the project length except of CPM. One of them (which can be easily extended to multi-criteria problems) is to find a longest path of the network. Unfortunately, this method cannot be adapted to solve (DM-Fix) since our objective function is not only dependent on the longest path, but on the delays of all paths from s to t. On a

first glance, the same holds for the *shrinking method* (Bein et al. 1992) of project planning, but in the following we will show that this approach can be used to solve (DM-Fix). In Section 4 we will then extend the shrinking method to solve (BDM).

We first sketch the shrinking method for minimizing the project length. It is based on the following operations:

Serial merge: Let a_1 be an arc from i to j, let a_2 be an arc from j to k, and let no other arc in $\bar{\mathcal{A}}$ be incident with node $j \in \bar{\mathcal{E}}$. Then merge a_1 and a_2 to one arc a from i to k with length $\bar{L}_a = \bar{L}_{a_1} + \bar{L}_{a_2}$.

Parallel merge: Let $a_1, \ldots, a_p$ be p arcs from i to j. Then merge all of them to one arc a from i to j with length $\bar{L}_a = \max\{\bar{L}_{a_1}, \ldots, \bar{L}_{a_p}\}$.

Node reduction: Let $i \in \bar{\mathcal{E}}$ such that only one incoming arc a_0 is incident with i, and let $a_1, \ldots, a_p$ be the outgoing arcs (or vice versa). Then merge a_0 with each arc $a_1, \ldots, a_p$ serially to obtain new activities $a_1^0, \ldots, a_p^0$; the length of each new activity is calculated as in the serial merge.

Any project network can be reduced to one single arc from s to t by applying these three operations. This final arc contains the minimal duration of the complete project. If the network is a series-parallel network, it can be reduced to one single arc by applying serial and parallel merges only (i.e. node reduction is not required).

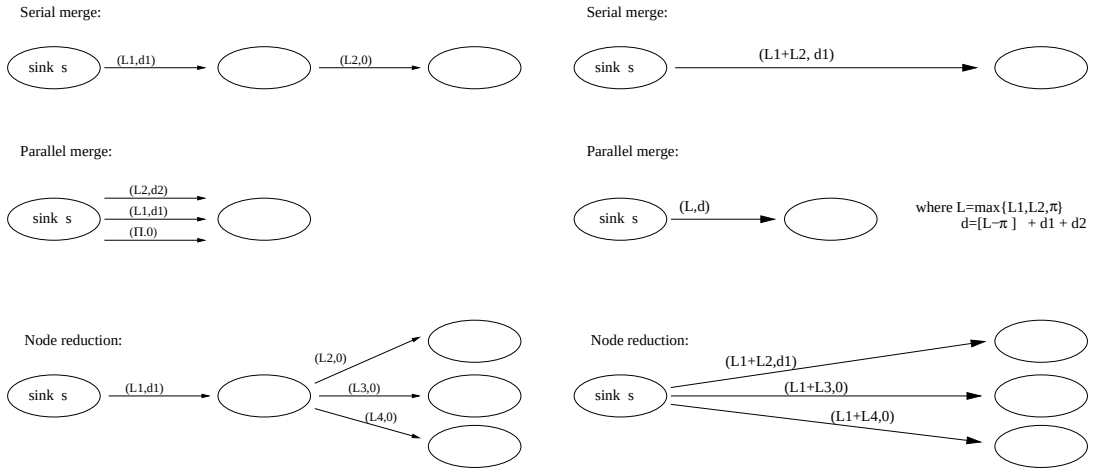

Figure 4: Situation before and after a serial merge, parallel merge, or node reduction, when calculating the delay.

We now adapt the shrinking method in such a way that we are able to calculate the delay correctly in the delay management problem. In order to keep track of

the delay in each of the operations we introduce a second parameter $\bar{d}_a$ for each activity. We furthermore shrink the network in a specific order, namely, we only merge activities if no preceding activities exist. This is specified in Algorithm 1. Note that during the reduction process parallel edges may occur, such that the notation $a = (i, j)$ means *activity a goes from event i to event j*. Furthermore, recall that each node (except of the source s and the sink t) is incident with exactly one incoming timetable arc.

Algorithm 1 for solving (DM-Fix)

Initialize: Set $\bar{d}_a = 0$ for each arc, and let $\bar{L}_a$ be defined as in equation (8).

Step 1: Until no further reduction is possible, do (see Figure 4)

Serial merge: Applicable for two activities $a_1 = (s, i)$, $a_2 = (i, j)$, $i, j \in \bar{\mathcal{E}}$ if no other activity is incident with i. Then delete i and merge a_1 and a_2 to a new arc $a = (s, j)$ with

$$\bar{L}_a = \bar{L}_{a_1} + \bar{L}_{a_2} \tag{9}$$
$$\bar{d}_a = \bar{d}_{a_1} \tag{10}$$

Parallel merge: Applicable for activities $a_1, \ldots, a_p$ with $a_k = (s, i)$ for a node $i \in \bar{\mathcal{E}}$ if there is no other activity with tail i.

Let a^s be the (at most one) timetable arc, i.e., $\bar{L}_{a^s} = \Pi_i$, $\bar{d}_{a^s} = 0$. Then merge $a_1, \ldots, a_p$ to one arc $a = (s, i)$ with

$$\bar{L}_a = \max\{\bar{L}_{a_1}, \ldots, \bar{L}_{a_p}\} \tag{11}$$
$$\bar{d}_a = \begin{cases} \sum_{a=a_1,\ldots,a_p} \bar{d}_a + (\bar{L}_a - \bar{L}_{a^s}) & \text{if } i \neq t \\ \sum_{a=a_1,\ldots,a_p} \bar{d}_a & \text{if } i = t \end{cases} \tag{12}$$

Node reduction: Applicable for activities $a_0, a_1, \ldots, a_p$ if $a_0 = (s, i)$ and $a_k = (i, j_k)$ with $i, j_k \in \bar{\mathcal{E}}$ for $k = 1, \ldots, p$, and no other activities are incident with i. Then arbitrarily choose an outgoing arc of i, say a_1, delete i and define the merged activities $a_1^0, \ldots, a_p^0$ with parameters

$$\bar{L}_{a_k^0} = \bar{L}_{a_0} + \bar{L}_{a_k}, k = 1, \ldots, p \tag{13}$$
$$\bar{d}_{a_1^0} = \bar{d}_{a_0} \tag{14}$$
$$\bar{d}_{a_k^0} = 0, \quad k = 2, \ldots, p \tag{15}$$

Step 2 : If the network has been reduced to one single arc $a = (s, t)$ let $f_{\mathcal{E}} = \bar{d}_a$. STOP.

Before proving the correctness of the algorithm, we present an example.

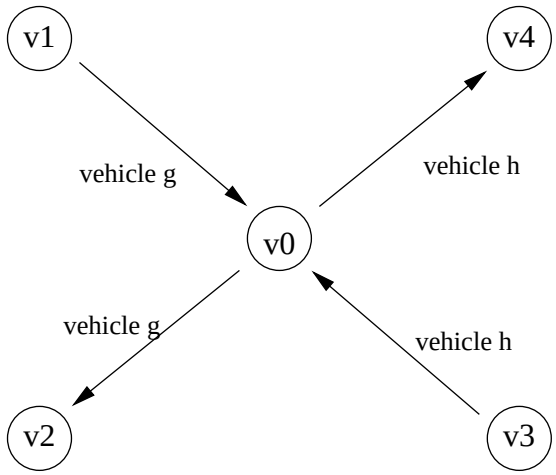

Figure 5: A PTN with two vehicles g and h.

Example 1 *Consider the situation shown in Figures 2, 5, and 6, consisting of two vehicles g and h that meet at a station v_0 where passengers can change between the two vehicles.*
The planned timetable and the lower bounds for each activity are given. Now let vehicle g arrive at station v_0 with a delay of 10 minutes, i.e., $\bar{d}_{s1} = 10$. The duration of the corresponding driving arc is $\bar{L}_{s1} = 18 + 10 = 28$. Figure 6 shows the complete network for the example, according to (6), (7), and (8), see page 10. Table 1 shows the disposition timetable y_i, the old timetable Π_i, and the delay $y_i - \Pi_i$ for all i. The sum of all delays in the network is 23 minutes which is minimal if both connections are maintained.

node	event	$\bar{y}_i$	Π_i	delay [min]
s		8:00	8:00	0
1	g, v_0, arr	8:28	8:18	10
2	h, v_3, dep	8:06	8:06	0
3	h, v_0, arr	8:20	8:20	0
4	g, v_0, dep	8:32	8:26	6
5	h, v_0, dep	8:34	8:27	7
6	g, v_2, arr	8:46	8:46	0
7	h, v_4, arr	8:47	8:47	0

$\sum_{i \in \mathcal{E}} (y_i - \Pi_i) = 23\,\text{min}$

Table 1: New timetable computed by CPM.

The following three lemmas show the correctness of Algorithm 1.

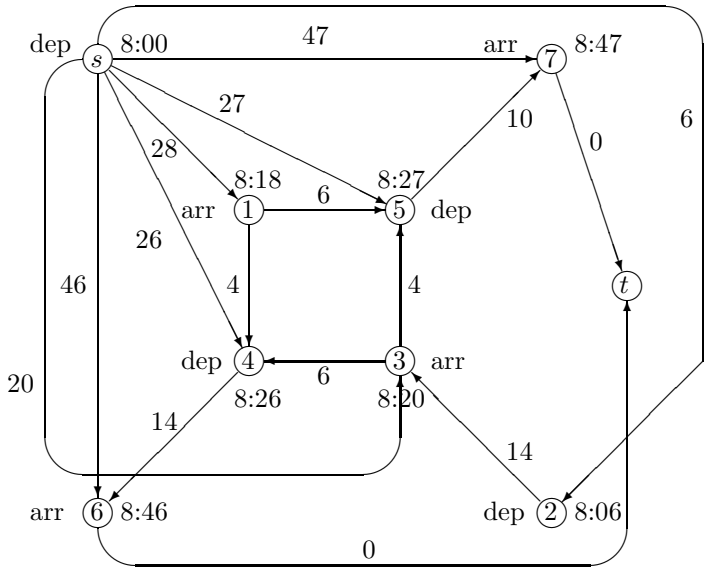

Figure 6: The PTN of Figure 5 as network $\bar{\mathcal{N}}$. Activity (s, i) is delayed by 10 minutes.

Lemma 2 *$\bar{\mathcal{N}}$ can be reduced to one single arc from s to t by Algorithm 1..*

Proof: Let $\bar{\mathcal{N}}^c = (\bar{\mathcal{E}}^c, \bar{\mathcal{A}}^c)$ denote the set of nodes occurring in the network after c steps of reduction. Note that there exists a directed path from s to each node in $\bar{\mathcal{E}}^c$, and from each node in $\bar{\mathcal{E}}^c$ to t. Take an arc $a_1 = (s, i)$ from s to the earliest node of the current network, i.e., find $i \in \bar{\mathcal{E}}^c$ such that no node $j \in \bar{\mathcal{E}}^c \setminus \{s\}$ exists with $j \prec i$.

- Either there are other arcs $a = (s, i)$, then a parallel merge is applicable.
- Otherwise, a_1 is the only incoming arc to event i. In this case, either $i = t$ (then $\bar{\mathcal{N}}^c$ consists of one arc only), or $i \neq t$. In the latter case there exists at least one arc starting at i, such that
  - a serial merge is applicable, or
  - a node reduction of node i can be performed.

QED

Lemma 3 *Let during the reduction process $a = (s, i)$ be the only activity from s to i, $i \in \bar{\mathcal{E}}$. Then L_a contains the disposition timetable for event i in the optimal solution of (DM-Fix).*

Proof: Applying the three shrinking operations together with the rules (9),(11), and (13) is equivalent to CPM (Elmaghraby 1977). Furthermore, CPM yields an optimal solution to (DM-Fix) (see Theorem 2). Together, the result follows.

QED

Lemma 4 *Let $a = (s, t)$ be the only remaining activity at the end of the shrinking process. Then $f_{\mathcal{E}} = \bar{d}_a$.*

Proof: We show by induction that in each step of the shrinking process $\sum_{a \in \bar{\mathcal{A}}} \bar{d}_a$ contains the sum of all delays of events in the set $\tilde{\mathcal{E}}$, consisting of

- events which have already been deleted during the process,
- *timetable events*, i.e., events $i \in \bar{\mathcal{E}}$ such that there exists exactly one incoming arc $a = (s, i)$.

After the initialization, we have $\bar{d}_a = 0$ for all $a \in \bar{\mathcal{A}}$. Since no activity has been deleted, $\tilde{\mathcal{E}}$ only contains events with no incoming activity in $\bar{\mathcal{A}}$, i.e., before the first delayed activity occurs. Hence, the claim is correct in this case. Now we discuss each of the three shrinking operations.

Serial merge of a timetable arc $a_1 = (s, i)$ and another arc $a_2 = (i, j)$ means to delete event i. Since i has been a timetable event before, and there exists at least one more arc to j (namely the timetable arc (s, j)) $\tilde{\mathcal{E}}$ does not change. On the other hand, activity a_1 is replaced by a, but with $\bar{d}_a = \bar{d}_{a_1}$, justifying the claim.

Node reduction consists of d serial merges. Since the delay of the timetable arc is transfered to exactly one of the new arcs, the claim stays correct under this operation.

Parallel merge of $a_1, \ldots, a_p$, all of them from s to i means that i becomes a timetable event and is added to $\tilde{\mathcal{E}}$. We have to show that the new calculation of the delay includes the delay of event i. First, consider the case that $i \neq t$ and a_1 is the unique timetable arc. Hence, $L_{a_1} = \Pi_i$ while $\bar{L}_a = \max_{k=2,\ldots,p} \bar{L}_{a_k}$ contains the (updated) time for event i in an optimal solution of (DM-Fix) due to Lemma 3. This means, the delay of event i is zero if $\bar{L}_a \leq \Pi$, otherwise the delay is given by $\bar{L}_a - \Pi_i$. Adding this new delay to $\bar{d}_{a_k}, k = 1, \ldots, p$ and using the induction hypothesis proves the result.

In the case that $i = t$ no further delay needs to be added. The claim stays correct, when replacing $a_1, \ldots, a_p$ by one arc a since all single delays are added to the delay of the new arc a.

QED

Lemma 3 and 4 together finally show the desired correctness of our procedure.

Theorem 3 *Algorithm 1 yields an optimal solution to (DM-Fix).*

Note that using the results of this section (Theorem 2 or 3) the two lexicographical minimal solutions of (BDM) can easily be determined:

- To minimize the number of missed connections $f_{\mathcal{A}}$ we define $\mathcal{A}_{fix} := \mathcal{A}_{change}$ and determine $f_{\mathcal{E}}$ by (CPM-F).
- To minimize $f_{\mathcal{E}}$ we proceed in two steps: First let $\mathcal{A}_{fix} = \emptyset$ and determine $f_{\mathcal{E}}$ by solving (DM-Fix) for no fixed connection. This yields a timetable y. To get the correct value of $f_{\mathcal{A}}$ we we have to check if $y_i - y_j < L_{ij}$ for each $(i,j) \in \mathcal{A}_{change}$.

Both lexicographic solutions are efficient. In the next section we present an approach which determines *all* efficient solutions of (BDM).

4 An exact algorithm for (BDM)

In this section we present an algorithm which finds the set of all efficient solutions of the bicriteria delay management problem (BDM). The algorithm is an extension of a procedure by (Demeulemeester et al. 1996) for the discrete time/cost trade-off problem (DTCTP). It is based on the shrinking method introduced in the previous section.

Let us start by first sketching the DTCTP and its solution procedure. The goal in the *discrete time/cost trade-off problem (DTCTP)* is to minimize the project length of a given project network, but one has the possibility to shorten the minimal duration of an activity $a \in \bar{\mathcal{A}}$ by spending additional money. The duration $\bar{L}_a$ is given as a **discrete** non-increasing function g_a depending on some cost c. The possible cost-duration combinations of the respective activities are called *modes* and are given by $\{(\bar{L}_a^m, c_a^m), m = 1, \ldots, M_a\}$. The goal of (DTCTP) is to find efficient solutions with respect to the following two criteria:

- minimize the project length, and
- minimize the costs.

While continuous cost-duration functions g_a have been widely studied (Elmaghraby 1977; Neumann 1975) and references therein, the literature on finding efficient solutions in the case of a discrete cost-duration function is rather sparse. (Demeulemeester et al. 1996) suggest to solve the problem with the following two procedures. The first algorithm is based on a procedure by (Bein et al. 1992) for finding the minimal number of reductions necessary to transform a general network to a series-parallel network. The second one enumerates alternative modes through a branch-and-bound tree. The approach has been further developed in

(Demeulemeester et al. 1998). It is based on the shrinking method, calculating the complete set of modes in each shrinking operation. In a serial or a parallel merge this can be done as follows.

Serial merge: $\{(\bar{L}_{a_1}^{m_1} + \bar{L}_{a_2}^{m_2}, c_{a_1}^{m_1} + c_{a_2}^{m_2}),$ for all modes m_1 of a_1, m_2 of $a_2\}$

Parallel merge: $\{(\max_{l=1,\ldots,p} \bar{L}_{a_l}^{m_l}, \sum_{l=1}^{p} c_{a_l}^{m_l})$ for all modes m_l of $a_l, l = 1, \ldots, p\}$

Note that many of the modes of the new arc a need not be constructed since they are dominated by other modes of a.

Unfortunately, the third of the three shrinking operations, namely node reduction, cannot be performed as easily, since one has to exclude the combination of two different modes of the same activity later on. Hence, for finding all efficient solutions one needs to fix a mode in each node reduction and to enumerate all possible combinations of modes. How to find a minimal set of activities to fix is described in (Bein et al. 1992). For further details, we again refer to (Demeulemeester et al. 1996).

We now turn our attention to the bicriteria delay management problem. To model (BDM) as a discrete time/cost trade-off problem we first have to define modes for each arc. For all activities $a \in \mathcal{A}_{change}$ we define two modes, given by $(\bar{L}_a, 0)$ and $(-\infty, w_a)$. Choosing the first mode means to maintain the connection, i.e., the duration of the changing arc has to be included in the calculation. On the other hand, choosing the second mode means that we do not maintain the connection and hence loose w_a passengers. All other activities (waiting, driving) as well as the timetable arcs only get one single mode $(\bar{L}_a, 0)$. The third parameter, calculating the delay in each step according to Section 3 is also necessary and hence included in the modes. Consequently, the modes are given by (duration, delay, weight of lost connections). In each step we reduce the network as in Algorithm 1. This can easily be done for serial and parallel merges, but if we perform a node reduction of some node i we have to distinguish two cases.

Case 1: If the common arc $a = (s, i)$ only has one mode, we proceed as in Algorithm 1.

Case 2: If $a = (s, i)$ has two or more modes, we fix one of the modes and store the remaining ones for further investigation.

Note that most of the activities only have one mode such that in many node reduction steps no activity has to be fixed and no additional branching arises.

Algorithm 2 for solving (BDM)

0. Initialize: Set $\bar{d}_a = 0$ for each arc, and let $\bar{L}_a$ be defined as in equation (8). Initialize one mode $m = (\bar{L}_a, \bar{d}_a, 0)$ for all $a \notin \mathcal{A}_{change}$ and two modes $m_1 = (L_a, \bar{d}_a, 0), m_2 = (-\infty, \bar{d}_a, w_a)$ for all $a \in \mathcal{A}_{change}$.

Step 1: Until none of the following operations is possible, do:

1.1. Serial merge: Applicable for two activities $a_1 = (s, i)$, $a_2 = (i, j)$, $i, j \in \bar{\mathcal{E}}$ if no other activity is incident with i. Then delete i and calculate the modes of the new activity a by combining each possible combination of modes, i.e.,

$$\{(\bar{L}_{a_1}^{m_1} + \bar{L}_{a_2}^{m_2}, \bar{d}_{a_1}^{m_1}, \bar{w}_{a_1}^{m_1} + \bar{w}_{a_2}^{m_2}), \text{ for all modes } m_1 \text{ of } a_1, m_2 \text{ of } a_2\} \quad (16)$$

where dominated modes are deleted.

1.2. Parallel merge: Applicable for activities $a_1, \ldots, a_p$ with $a_k = (s, i)$ for one common node $i \in \bar{\mathcal{E}}$ if there is no other incoming activity of event i.

Let a^s be the (at most one) timetable arc. Then the modes m_l of the new activity a_l are given by

$$\{(\max_{l=1}^{p} \bar{L}_{a_l}^{m_l}, \sum_{l=1}^{p} \bar{d}_{a_l}^{m_l} + (\max_{l=1}^{p} \bar{L}_{a_l}^{m_l} - \bar{L}_{a^s}), \sum_{l=1}^{p} \bar{w}_{a_l}^{m_l}) \text{ for all } l = 1, \ldots, p\}$$

in the case that $i \neq t$. For $i = t$ the modes are given by

$$\{(\max_{l=1}^{p} \bar{L}_{a_l}^{m_l}, \sum_{l=1}^{p} \bar{d}_{a_l}^{m_l}, \sum_{l=1}^{p} \bar{w}_{a_l}^{m_l}) \text{ for all } l = 1, \ldots, p\}$$

where again, dominated modes are deleted.

1.3. Node reduction: Applicable for activities $a_0, a_1, \ldots, a_p$ if $a_0 = (s, i)$ only has one single mode, $a_k = (i, j_k)$ with $i, j_k \in \bar{\mathcal{E}}$ for $k = 1, \ldots, p$, and no other activity is incident with i. Then arbitrarily choose an outgoing arc of i, say a_1, delete i, calculate the modes of the new activity $a_k^0, k = 1, \ldots, p$ by

$$\{(\bar{L}_{a_0} + \bar{L}_{a_k}^{m}, \bar{d}_{a_0}, \bar{w}_{a_0} + \bar{w}_{a_k}^{m}) \text{ for all modes } m \text{ of } a_k\}, \quad (17)$$

and delete dominated modes.

Step 2 :

2.1 Final reduction: If the network has been reduced to one single arc goto 3.

2.2 Fixing an activity: Otherwise, choose activity $a = (s, i)$ such that i is the earliest node still in the network, fix one mode of activity a and perform node reduction as in step 1.3. Goto step 1.

Step 3: Evaluate: Add the modes of the new solution, delete all dominated ones (compared to other solutions obtained in previous steps) and start again at step 1, fixing another combination of modes during step 2.2

If all combinations of modes have been calculated, STOP.

Theorem 4 *Algorithm 2 finds all efficient solutions of (BDM).*

Proof: Using the result of Theorem 4 we know that the delay $f_{\mathcal{E}}$ has been calculated correctly when reaching step 2.1. Furthermore, the final value of $\bar{w}$ in the remaining activity from s to t equals $f_{\mathcal{A}}$. If all solutions had been determined for each possible combination of modes, the resulting set would contain all efficient solutions. Since a solution can never be Pareto if parts of it are dominated (i.e., can be replaced by a better solution) it is feasible to delete dominated modes for single activities during the reduction process. Since in the final step, all remaining dominated solutions are deleted, we end up with the set of non-dominated solutions.

QED

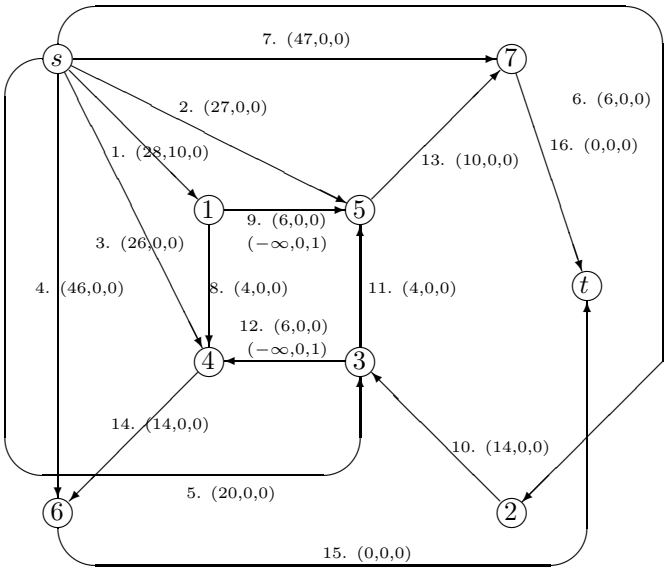

Figure 7: $\bar{\mathcal{N}}$ with the modes for DTCTP.

Example 2 *Consider again Example 1. Figure 7 shows this example as a DTCTP. First, a serial merge operation can be applied to activities 6 and 10. We denote the new activity with mode (20,0,0) as number 17 and merge it in parallel with activity 5 which yields the new activity 18 with mode (20,0,0). Now a node reduction step is performed for event 1, adding the costs of activity 1 to a succeeding activity (in our example activity 8). We obtain activity 19 with mode (32,10,0). Activity 9 can then be merged serially with activity 1. (Note that the costs are not added in this case to avoid double counting.) This yields activity 20 with modes (34,0,0) and (−∞,0,1). Now merge activity 3 and 19 in parallel. Since activity 3 is a timetable arc and the duration of activity 19 is greater than the one of*

activity 3, the difference of 6 minutes has to be added to the delay of the new activity, which is numbered by 21 and has mode (32,16,0). Another parallel merge operation can be performed with activities 2 and 20. It results in activity 22 with modes (34,7,0),(27,0,1). The delay of 7 minutes in the first mode results from the difference in the duration of activities 20 and 2 ($34-27=7$). Figure 8 shows the network after these reduction steps. We can continue in the same way by performing a node reduction with event 3. Table 2 shows the complete reduction plan.

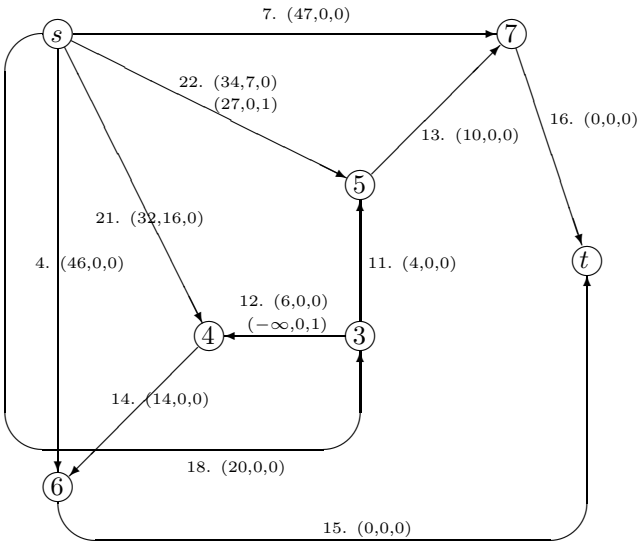

Figure 8: Network after the first six reduction steps.

The EVALUATE *construct in step 18 is used to compare the obtained solution with solutions found in previous iterations. Since in this example we did not fix an activity with more than one mode, the computation can be stopped and yields two solutions: the first one with no missed connection and 23 minutes of total delay and the second one with one missed connection, but only 16 minutes of total delay.*

5 Numerical results using real-world data

The algorithm described in Section 4 was implemented in C++ using LEDA[1]. The input data was provided by a large traffic association *Verkehrsverbund Rhein-Neckar* operating in the south-west of Germany. In the test data we analyze the

[1]Library of Efficient Data types and Algorithms.

Action	Activities	New	Modes
1.Series	[6,10]	17	(20,0,0)
2.Parallel	[5,17]	18	(20,0,0)
3.Reduce	[1,8]	19	(32,10,0)
4.Reduce	[1,9]	20	(34,0,0),$(-\infty$,0,1)
5.Parallel	[3,19]	21	(32,16,0)
6.Parallel	[2,20]	22	(34,7,0),(27,0,1)
7.Reduce	[18,11]	23	(24,0,0)
8.Reduce	[18,12]	24	(26,0,0),$(-\infty$,0,1)
9.Parallel	[21,24]	25	(32,16,0)
10.Parallel	[22,23]	26	(34,7,0),(27,0,1)
11.Series	[25,14]	27	(46,16,0)
12.Parallel	[4,27]	28	(46,16,0)
13.Series	[26,13]	29	(44,7,0),(37,0,1)
14.Parallel	[7,29]	30	(47,7,0),(47,0,1)
15.Series	[28,15]	31	(46,16,0)
16.Series	[30,16]	32	(47,7,0),(47,0,1)
17.Parallel	[31,32]	33	(47,23,0),(47,16,1)
18.Evaluate	33		

Table 2: Reduction plan for the example.

effects of delays of trains on a special line (called *Lautertalbahn*) on subsequent other trains and buses belonging to different public transportation companies. The test set consists of

- 823 stations,
- 1314 vehicles, and
- 2118 direct rides.

As connections we considered all transfers possible within a time interval of 30 minutes. We obtained nearly 40.000 changing activities. The event-activity network of a complete day hence consists of roughly 45.000 events and 85.000 activities. All computations were done on a PC with a 266 MHz Intel Pentium II processor.
The program we implemented allows to specify a time interval to select the events which should be taken into account. This time interval in minutes is given in the first column of the tables. Depending on the time period considered, the number of events and activities can be drastically reduced, such that the resulting (relevant) part of the network typically consists of roughly 2.000 events and 4.000 activities. The second column contains the number of changing activities which may be affected. Moreover, the amount of the source delay, the CPU time in seconds and the number of efficient solutions found by the program are given.

Example 3 *Table 3 shows an example from the "Lautertalbahn". A train from station "Untersulzbach" to "Hirschhorn" arrives at its destination with delay. Since a larger delay is not so easily compensated by the slack times it can spread out through the network and hence yields a larger number of efficient solutions as a small delay.*

Table 4 summarizes the results we obtained in our numerical studies (Ginkel 2001). Each row contains the results belonging to different delayed activities, which could typically occur in practice. For each delayed event, at least three different amounts of source delays between 10 and 20 minutes have been tested. The last two time intervals, 60 and 70 minutes, have been computed only for one delayed activity, but with different amounts of source delay. The table shows that the behavior of the algorithm and the number of efficient solutions found strongly depends on the respective problem instance.
Summarizing, our numerical experiments indicate that Algorithm 2 runs rather quickly if the relevant time interval to be considered is not too large. Evaluating the delay and the missed connections for all events and activities within the next 60 minutes could in most cases be done within a running time of less than a minute on a standard personal computer. Both the running time and the number of efficient solutions depend on the number of changing activities that need to be considered. This behavior is depicted in Figure 9.

time interval in [min]	changing arcs affected	source delay in [min]	cpu time in [sec]	no. of efficient solutions
20	1	10	0.00	2
		15	0.01	2
30	9	10	0.04	4
		15	0.05	6
35	20	10	0.45	6
		15	0.72	10
40	47	10	28.93	6
		11	68.40	9
		12	99.12	10
		13	144.64	13
		14	145.94	13
		15	187.62	15

Table 3: Example with different amounts of the source delay.

time interval in [min.]	changing arcs	CPU time in [sec.]	no. of efficient solutions
20	[0,2]	[0.00, 0.01]	[1,3]
30	[0,9]	[0.01, 0.05]	[1,6]
40	[1,47]	[0.01, 187.6]	[2, 15]
50	[17,25]	[0.37, 15.93]	[2, 14]
60	28	[1.17, 22,56]	[2,7]
70	45	[3.17, 79.07]	[2,7]

Table 4: Numerical results for some real-world delays.

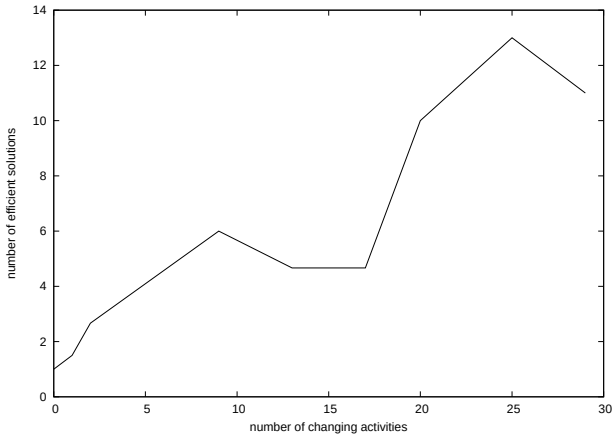

Figure 9: Number of efficient solutions as function of the size of $\mathcal{A}_{change}$.

The nice behavior of Algorithm 2 on our data is due to the fact that the number of changing activities outside of the municipal areas is relatively small, such that the event-activity network is close to a series-parallel one.

6 Conclusions and further research

We presented a new model treating the delay management problem in a bicriteria setting, and developed a solution technique similar to the discrete time/cost tradeoff problem in project planning. The method seems to have the potential to be used as an online decision support procedure. Improvements can be obtained by applying a branch and bound approach instead of fixing all possible combinations of modes.

In delay management, the following two extensions seem to be most relevant for solving real-world problems. The first is to include the limited capacity of railway systems in the models, making sure that the wait-depart decisions lead to a disposition timetable that is realizable without any conflicts. (An iterative procedure for this problem is currently investigated within the project *DisKon* together with *Deutsche Bahn*.) Another interesting topic is the question of robustness of disposition timetables in the delay management problem. This question is under research within the European research project ARRIVAL.

References

T. Ackermann. *Die Bewertung der Pünktlichkeit als Qualitätsparameter im Schienenpersonennahverkehr auf Basis der direkten Nutzenmessung.* PhD

thesis, Universität Stuttgart, 1999.

N. Bissantz, S. Güttler, J. Jacobs, S. Kurby, T. Schaer, A. Schöbel, and S. Scholl. DisKon - Disposition und Konfliktlösungs-management für die beste Bahn. *Eisenbahntechnische Rundschau (ETR)*, 45(12):809–821, 2005. (in German).

W. Bein, J. Kamburowski, and M. Stallmann. Optimal reduction of two-terminal directed acyclic graphs. *SIAM Journal on Computing*, 21(6):1112–1129, 1992.

E. Demeulemeester, W. Herroelen, and S. Elmaghraby. Optimal procedures for the discrete time/cost trade-off problem in project networks. *European Journal of Operational Research*, 88:50–68, 1996.

E. Demeulemeester, B. De Reyck, B. Foubert, W. Herroelen, and M. Vanhoucke. New computational results on the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, 49:1153–1163, 1998.

M. Ehrgott. *Multiple Criteria Optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Berlin, 2005.

S.E. Elmaghraby. *Activity Networks*. Wiley Interscience Publication, 1977.

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.

L. Giovanni, G. Heilporn, and M. Labbé. Optimization models for the delay management problem in public transportation. *European Journal of Operational Research*, 2006. to appear.

A. Ginkel. Event-activity networks in delay management. Master's thesis, Universität Kaiserslautern, 2001.

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

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

C. Liebchen and S. Stiller. Delay resistant timetabling. Technical Report 2006/24, Technische Universität Berlin, 2006. presented at CASPT'06.

C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller, and A. Prigge. Computing delay-resistant railway timetables. Technical report, Institut für Numerische und Angewandte Mathematik, Georg-August Universität Göttingen, 2007. ARRIVAL Report TR-0066.

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

K. Neumann. *Operations Research Verfahren*, volume III. Carl Hanser Verlag, München Wien, 1975.

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.

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.

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

A. Schöbel. *Customer-oriented optimization in public transportation.* Optimization and Its Applications. Springer, New York, 2006.

A. Schöbel. Integer programming approaches for solving the delay management problem. *Lecture Notes in Computer Science*, 2007. to appear.

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.

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.

A. Schöbel and S. Schwarze. A game-theoretic approach to line planning. In *6th workshop on algorithmic methods and models for optimization of railways*, number 06002 in Dagstuhl Seminar proceedings, 2006.

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.