Erasmus School of Economics

Final version

An Evaluation Study on Online-Strategies for Delay Management

Author:
R. Sewnarain

Supervisor:
T. Dollevoet

Contents

1 Introduction

The *delay management problem* asks how to deal with (small) *exogenous delays* of a railway system, while minimizing the total passenger delay. Such *source delays* usually make the *scheduled timetable* infeasible: From the moment the delay occurs, not all operational constraints can be met. Therefore, the infeasible scheduled timetable has to be updated to a feasible *disposition timetable*. So far, two main aspects are treated in the literature. The first involves a *wait-depart decision*, where the question arises which trains should wait for delayed feeder trains and which trains better depart on time. This leads to the *uncapacitated delay management problem* (UDM) (see Schachtebeck and Schöbel [2]). The second involves a priority decision regarding the limited capacity of the track system. Each time two trains compete for the same part of the railway infrastructure, it has to be decided which train may go first. *Headway constraints* model this limited capacity (see Schachtebeck and Schöbel [2]). The delay management problem is further complicated by its online nature. Exogenous delays in public railway traffic are usually not known in advance and therefore decisions have to be made without exactly knowing the future. Bauer and Schöbel [1] focused on this *online delay management*. They enhanced both offline models proposed in Schachtebeck and Schöbel [2] for the online case. As a result, they gained a generic model that is able to cover complex realistic memoryless delay scenarios. Furthermore, they introduced and evaluated online strategies for delay management that are practical, easily applicable and robust. To do this, they stated three approaches for solving the problem at hand: An ILP-based approach, a class of simple 'rule of thumb' strategies and a learning heuristics that makes use of simulation. They found that the ILP-based and the learning heuristics perform very well, usually resulting in near-optimal solutions. The goal of my research is to determine whether these results still hold for Dutch railway data by implementing two of the three approaches proposed by Bauer and Schöbel [1]. As the ILP-based and the learning heurstics are the most promosising, we chose to implement these. We will do this for the uncapacitated case which means that we refrain from headway constraints. We do this because our time is limited during this project. Hopefully this will lead to interesting results.

2 Problem Statement

In this section we will introduce notation to define the problem at hand. Furthermore, some definitions are given to grasp the *delay management process* with which we can model *offline* and *online delay management*.

2.1 Notation

Before we can model the railway system, we need to introduce some notation. An *event-activity* network is a directed graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$. The nodes in $\mathcal{E}$ are decomposed into arrival and departure events, $\mathcal{E}_{arr} \cup \mathcal{E}_{dep}$, and correspond to trains. The edges in $\mathcal{A}$ are called *activities* and are decomposed into $\mathcal{A} = \mathcal{A}_{drive} \cup \mathcal{A}_{wait} \cup \mathcal{A}_{change}$, with each activity $a \in \mathcal{A}$ having a minimal duration $L_a > 0$. Note that this differs with Bauer and Schöbel [1] as we refrain from *headway* activities. A headway activity models the limited capacity of the railway system. This can either be two trains driving on the same track into the same direction or two trains driving into opposite directions on a single-way track. The meaning of the remaining activities is as follows:

- A *driving* activity represents a train driving between two consecutive stations.
- A *waiting* activity corresponds to the time period in which a train is waiting at a station to let passengers on or off.
- A *changing* activity corresponds to the transfer of passengers from one train to another.

For each changing activity a decision has to be made whether it stays in the network or it should be deleted. If the activity $(i, j) \in \mathcal{A}_{change}$ stays in the network this means that either the follow-up train (corresponding to node j) will wait for the passengers that have to be transferred from the delayed feeder train (corresponding to node i) or the feeder train has no delay and the activity (i, j) can be maintained anyway. As there is no decision to be made for driving and waiting activities we abbreviate $\mathcal{A}_{wait} \cup \mathcal{A}_{drive}$ by $\mathcal{A}_{train}$.

2.2 Modelling Offline Delay Management

A *delay state* d reflects the current knowledge and expectations on exogenous delays in the underlying railway system. These delays are called *source delays*. Such delays are for example technical problems with the railway infrastructure or the late arrival of a train driver. *Offline delay management* assumes that all source delays are known beforehand. In reality this is of course not the case as we do not know when an exogenous delay will occur. We use $d_i \in \mathbb{R}_+$ and $d_{(i,j)} \in \mathbb{R}_+$, which means there is a source delay of d_i minutes at event i and a source delay of $d_{(i,j)}$ minutes at activity (i, j), respectively. When the source delays alter, the scheduled timetable π has to be updated to a disposition timetable x. We will now give the definition of a *timetable* and a *disposition timetable*.

Timetable. A timetable for $\mathcal{N}$ is a vector $x \in {\mathbb{R}_+}^{|\mathcal{E}|}$ which assigns a time $x_i \in \mathbb{R}_+$ to each event $i \in \mathcal{E}$ such that:

$$x_j - x_i \geq L_{(i,j)} \qquad \forall (i, j) \in \mathcal{A}_{train}, \tag{1}$$

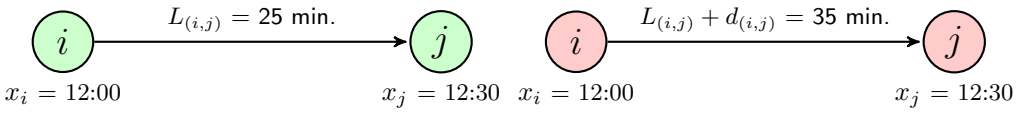
with $L_{(i,j)}$ being the minimal duration for activity (i, j). Equation (1) assures that event j cannot start before activity (i, j) has been finished. If each activity satisfies (1), we call the timetable *feasible*.

Disposition timetable. Given a delay state d, a disposition timetable $x \in \mathbb{R}_+^{|\mathcal{E}|}$ for d is a timetable for $\mathcal{N}$ such that:

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

$$x_j \geq x_i + d_{(i,j)} + L_{(i,j)} \qquad \forall (i, j) \in \mathcal{A}_{train} \tag{3}$$

Equations (2) and (3) take source delays into account and assure that the timetable is feasible. Figure 1 shows two situations in which the timetable is either feasible or infeasible for two events $i, j \in \mathcal{E}$ and activity $(i, j) \in \mathcal{A}_{train}$.



(a) feasible timetable (b) infeasible timetable

Figure 1: Example of a feasible and an infeasible timetable.

We write $z^x_{(i,j)} = 0$ if and only if the changing activity (i, j) is maintained by the disposition timetable x, and $z^x_{(i,j)} = 1$ otherwise. We can then calculate the *total delay* of a disposition timetable x:

$$f(x) = \sum_{i \in \mathcal{E}_{arr}} w_i(x_i - \pi_i) + \sum_{(i,j) \in \mathcal{A}_{change}} z^x_{(i,j)} w_{(i,j)} T, \tag{4}$$

where $w_i \in \mathbb{N}$ stands for the number of passengers getting off at event $i \in \mathcal{E}_{arr}$ and arriving at their final destination, and where $w_{(i,j)} \in \mathbb{N}$ stands for the number of passengers who make use of the transfer. At last, we interpret $T \in \mathbb{R}_+$ as a penalty for not maintaining the changing activity. If we express T in hours instead of minutes, then we can see $\frac{1}{T}$ as the arrival rate per hour of the train. More intuitively, if you miss your transfer, you should wait at least T minutes before you can take the next train.

We now have the following problem at hand: we have to find a disposition timetable x for delay state d where $f(x)$ is minimized.

Note that the graph $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ does not contain headway activities in contrast to Bauer and Schöbel [1], which means that $\mathcal{N}$ is acyclic. In order to construct a timetable for the directed graph $\mathcal{N}$, it needs to be sorted topologically. That means, for every activity $(s, t) \in \mathcal{A}$, s comes before t in the ordering. The topological ordering then gives a valid sequence of the nodes that has to be handled when constructing the timetable. A topological ordering is only possible if the graph has no directed cycles and any directed acyclic graph has at least one topological ordering. Therefore, π exists.

2.3 Modelling Online Delay Management

It is of course not realistic to assume that all delays are known beforehand. We would like to model unexpected delays such as accidents or suicidal people who jump in front of the train. Ergo, source delays are not completely known in advance. The knowledge and expectation on delays are given by a finite sequence of pairs $\sigma = (d^1, t^1), (d^2, t^2), ..., (d^m, t^m)$, where $d^k \in \mathbb{R}_+^{|\mathcal{A}_{train}|+|\mathcal{E}|}$ is a delay state and $t^k \in \mathbb{R}_+$ the time at which the delay state starts. Just like in Bauer and Schöbel [1], where they use a *delay generator* to generate the delay states, we assume that changing activities can not be delayed. Between time t^k and t^{k+1} everything happens as expected in delay state d^k. The following definitions are needed in order to grasp the *delay management process*.

Feasible next delay state. Given delay states d^1 and d^2, a timetable x^1 and a point in time $t^2 \in \mathbb{R}_+$, we call (d^2, t^2) feasible for (x^1, d^1) if $d^1_{(i,j)} = d^2_{(i,j)}$ for each activity (i, j) with $x^1_j < t^2$ and $d^1_i = d^2_i$ for each event with $x^1_i < t^2$.

As stated before, when the delay state alters, the current timetable can be infeasible and needs to be updated to a (feasible) disposition timetable. An *online strategy for delay management* computes a disposition timetable to fit the new delay state. When doing so, the topological ordering needs to be respected. However, we only want to schedule the events that may be influenced by the given delay state. Let us assume that the time now is t^2 and delay state d^2 occurs. Events which took place before t^2 already happened and therefore do not influence the new disposition timetable x^2. That is, for every $i \in \mathcal{E}$ with $x^1_i < t^2$ the delay state d^1_i is the *same* as d^2_i. The same holds for every activity $(i, j) \in \mathcal{A}$ with $x_j < t^2$.

Online strategy for delay management. Given are a timetable x^1, a time $t^2 \in \mathbb{R}_+$ and delay states d^1, d^2, such that (d^2, t^2) are feasible for (x^1, d^1). An *online strategy* for delay manegement is an algorithm $\mathcal{S}$ that computes a disposition timetable $x^2 = \mathcal{S}(x^1, d^2, t^2)$ for d^2 such that:

$$x_i^2 = x_i^1 \quad \forall i \in \mathcal{E} \quad \text{with} \quad x_i^1 < t^2 \tag{5}$$

$$x_i^1 \geq t^2 \qquad \text{implies} \qquad x_i^2 \geq t^2 \tag{6}$$

Events that already happened before the current time t^2, at which the timetable is being rescheduled, do not need to be scheduled. (5) assures this. (6) makes sure that events which can be influenced by the given delay state d^2 are being rescheduled. As they did not occur yet, they need to be scheduled to a time greater than or equal to t^2.

The current state of the system is expressed through the triple (x, d, t). Then, the next delay state and time (d', t') are randomly chosen by a *delay generator*.

Delay generator. Let $\mathcal{D}$ be a black-box routine whose input may consist of current disposition timetable x, delay state d, time t, additional random values and values computed in former executions of $\mathcal{D}$. We call $\mathcal{D}$ a *delay generator* if $(d', t') := \mathcal{D}(x, d, t)$ is a feasible next delay state for input (x, d, t) and any additional input.

Now that we familiarized ourselves with the necessary definitions, we are ready to define the *delay management process*.

Delay management process. We are given a delay management strategy $\mathcal{S}$ and a delay generator $\mathcal{D}$. The delay management process $\mathcal{M}$ with respect to $\mathcal{S}$ and $\mathcal{D}$ is the process constructed as follows: Starting with $x^0 = \pi$, $d^0 = 0$ and t^0, we iteratively obtain $(d^{i+1}, t^{i+1}) := \mathcal{D}(x^i, d^i, t^i)$ and afterwards compute $x^{i+1} = \mathcal{S}(x^i, d^{i+1}, t^{i+1})$. The process ends when $t^{m+1} = \infty$. Let $\sigma' = (x^1, d^1, t^1), (x^2, d^2, t^2), ..., (x^m, d^m, t^m)$ be a realization of $\mathcal{M}$. The *total delay* of σ' is $f(x^m)$

For online delay management we now have the following problem at hand: we have to find an online strategy $\mathcal{S}$, such that the expected overall delay of the delay process management process $\mathcal{M}$ of $\mathcal{S}$ and $\mathcal{D}$ is minimal.

Before we present two delay management strategies, we define the delay generator proposed by Bauer and Schöbel [1] that we used in our experiments.

Delay Generator: Delays Determined in Advance (DDA). Delays do *not* depend on the delay management strategy. Before the actual delay management strategy starts we generate an instance d^{final} of DDA. To do so, we need the number of *train* activities $n_{train} \leq |\mathcal{A}_{train}|$ and the number of events $n_{events} \leq |\mathcal{E}|$ that are source delayed. We then choose them randomly from the data and assign a random value $u \sim \text{UNI}(a,b)$ with $a, b \in \mathbb{N}, u \in \mathbb{R}_+$ to that event or activity in d^{final}, which denotes a source delay of u minutes. The delay management process starts with state $(x^0 = \pi, d^0 = 0, t^0 = 0)$.

Let $i \in \mathcal{E}$ be such that $d_i^k = 0$, $d_i^{final} > 0$ and x_i^k is minimal, and let $(j, w) \in \mathcal{A}$ be such that $d_{(j,w)}^k = 0$, $d_{(j,w)}^{final} > 0$ and x_j^k is minimal. We can then determine t^{k+1}, the time at which the next delay occurs and thus the time at which the timetable is being rescheduled. More formally, $t^{k+1} = \min\{x_i^k, x_j^k\}$. Given arbitrary $a \in \mathcal{E}, (v, w) \in \mathcal{A}_{train}$, the next delay state d^{k+1} is then given by:

$$d_a^{k+1} = \begin{cases} d_a^{final} & , x_a^k \leq t^{k+1} \text{ or } d_a^k > 0 \\ 0 & , \text{otherwise} \end{cases} \tag{7}$$

$$d_{(v,w)}^{k+1} = \begin{cases} d_{(v,w)}^{final} & , x_v^k \leq t^{k+1} \text{ or } d_{(v,w)}^k > 0 \\ 0 & , \text{otherwise} \end{cases} \tag{8}$$

3 Delay Management Strategies

In this section we will propose two online strategies: an *Online Mixed Integer Program* (OnlineMIP) based on a Offline Integer Linear Program (OfflineILP) proposed in Bauer and Schöbel [1] & Schachtebeck and Schöbel [2], and an *Ad-Hoc Re-Scheduling procedure* based on heuristic functions used in Bauer and Schöbel [1].

3.1 OnlineMIP

The OnlineMIP approach makes use of the offline problem. Schachtebeck and Schöbel [2] give an exact integer programming formulation for the offline problem. Note that in our case the schedule consists of events planned with time $x_i \in \mathbb{R}_+,\ \forall i \in \mathcal{E}$ instead of $x_i \in \mathbb{N},\ \forall i \in \mathcal{E}$ in Bauer and Schöbel [1]. Therefore, we have an OnlineMIP instead of an OnlineILP. Bauer and Schöbel [1] showed that when working with the delay generator DDA, the optimal solution of the offline problem gives an a-posteriori bound on the corresponding online problem of that instance (determined by d^{final}). We can use this to evaluate how good our solution is using an online strategy. The OnlineMIP can be used as an online strategy by iteratively recomputing the exact offline-solution of the past events. We will now give the MIP in order to solve the offline problem.

$$\underset{x,z}{\text{minimize}} \quad f(x,z) := \sum_{i \in \mathcal{E}} w_i(x_i - \pi_i) + \sum_{a \in \mathcal{A}_{change}} z_a w_a T \tag{9}$$

$$\text{s.t.} \quad x_i \geq \pi_i + d_i \qquad i \in \mathcal{E} \tag{10}$$

$$x_j - x_i \geq L_{(i,j)} + d_{(i,j)} \qquad (i,j) \in \mathcal{A}_{train} \tag{11}$$

$$M z_{(i,j)} + x_j \geq L_{(i,j)} + x_i \qquad (i,j) \in \mathcal{A}_{change} \tag{12}$$

$$x_i \in \mathbb{R}_+ \qquad i \in \mathcal{E} \tag{13}$$

$$z_a \in \mathbb{B} \qquad a \in \mathcal{A}_{change} \tag{14}$$

The constant M has been studied in Schachtebeck and Schöbel [2] and in our case, where we refrain from headway activities, $M = \max_{i \in \mathcal{E}} \{d_i + \sum_{a \in \mathcal{A}_{train}} d_a\}$ ensures that M is 'big enough'. We already discussed (9), (10) and (11). (12) is associated with the wait-depart decision. If the decision variable $z_{(i,j)} = 1$, this means that the follow-up train will not wait for the delayed feeder train.

We are now ready to define the OnlineMIP strategy.

OnlineMIP. Given (x^k, d^{k+1}, t^{k+1}), compute $x^{k+1} = \mathcal{S}(x^k, d^{k+1}, t^{k+1})$ by setting $x^k = x$ for a feasible solution of the ILP with additional constraints:

$$x_i = x_i^k, i \in \mathcal{E}, x_i^k < t^{k+1}$$
$$x_i \geq t^{k+1}, i \in \mathcal{E}, x_i^k \geq t^{k+1}$$

3.2 Ad-Hoc Re-Scheduling (AHRS)

Ad-Hoc Re-Scheduling is an online strategy which updates the timetable event-by-event. This strategy makes use of three functions. In order to update the timetable, the events need to be sorted topologically. The function $time_{\text{top}}$ makes sure this ordering is respected. Heuristic functions are used to decide the time and choice of the next event to schedule. In the function $time_{\text{dm}}$ the heuristic function can be specified which is associated with the decision whether the follow-up train should wait for the delayed feeder train. Bauer and Schöbel [1] uses a class of simple 'rule of thumb' strategies and a learning heuristics. Examples of such 'rule of thumb' strategies are for example to always wait for the delayed feeder train or to never wait. The learning heuristics is based on a simulation where the time how long a follow-up train should wait for a feeder train is 'learned' for each changing activity. We will describe this heuristic later in more detail. At last, the function $time_{\text{earliest}}$ assures that all technical restrictions are respected and makes sure we find a feasible timetale. Let us now describe the Ad-Hoc Re-Scheduling strategy in more detail.

Initialization. Let time t^{k+1} be the time at which the timetable needs to be updated. We then need to determine which events already happened (and thus need no re-scheduling) and which events can be influenced by the given delay state. Therefore, the scheduled times of events that already happened will remain the same, whereas events that need to be re-scheduled will get the value ∞ which indicates that those events still need to be planned.

$$time_{\text{top}}(i,x) = \begin{cases} \infty & \text{, there is an } (j,i) \in \mathcal{A}_{change} \text{ with } x_j = \infty \\ 0 & \text{, otherwise} \end{cases} \tag{15}$$

Time top. The definition of the function $time_{\text{top}}$ is given in equation (15). If we are planning event i, we need to determine if there are changing activities to node i. If so, we need to check whether this event is already planned. If this event is not planned yet, this means that scheduling event i does not respect the sorted topology and the function returns the value ∞. In any other case, the function returns the value zero which indicates that event i can be scheduled. It is clear now that this function ensures the topological ordering of the changing activities.

$$time_{\text{earliest}}(i,x,d,t) = \max(\{t, \pi_i + d_i\} \cup \{x_j + L_{(j,i)} + d_{(j,i)} | (j,i) \in \mathcal{A}_{train}\}) \tag{16}$$

Time earliest. For a definition of the function $time_{\text{earliest}}$ see equation (16). This function determines the earliest point in time at which event i can take place, taking into account the original timetable π, the predecessing event on the same line and the current time.

$$time_{\text{dm}}(i,x,d) = \max_{(j,i)\in\mathcal{A}_{change}} \{1_{[0,\pi_i + l_{(j,i)}]}(x_j + L_{(j,i)})\} \tag{17}$$

Time dm. The function $time_{\text{dm}}$ presented in equation (17) determines if a follow-up train should wait for a (delayed) feeder train. This function depends on the value $l_{(j,i)} \in \mathbb{N}$, meaning that event j should wait at most $l_{(j,i)}$ minutes starting from the scheduled timetable in order to maintain (i,j). To learn the values of $l_{(j,i)}$ we make use of the learning heuristics presented in Bauer and Schöbel [1].

Learning heuristics. The idea is as follows. First, a number of random delay states has to be generated. Afterwards, we solve the OfflineMIP on these instances. We obtain a sequence $\sigma'' = (x^1, d^1), ..., (x^n, d^n)$ where disposition timetable x^i is an optimal solution of the random delay state d^i. We are interested in how long event j should wait starting from the scheduled timetable in order to maintain changing activity (i,j). We denote this by $l_{(i,j)}$. We then calculate the multisets $YES_{(i,j)}$ and $NO_{(i,j)}$. These sets contain, for each instance, how long event j would have had to wait in order to maintain changing activity (i,j). The set YES consists of instances where the activity (i,j) is maintained. The set NO contains the remaining instances. At last, we let $l_{(i,j)}$ be an arbitrary value that seperates $YES_{(i,j)}$ and $NO_{(i,j)}$ optimally, i.e. the value $l_{(i,j)}$ for which (18) is minimal.

$$p(l_{(i,j)}) := |\{v \in YES_{(i,j)} | v > l_{(i,j)}\}| + |\{v \in NO_{(i,j)} | v < l_{(i,j)}\} \tag{18}$$

We can solve this for every changing activity by solving the following problem.

$$\underset{y,n}{\text{minimize}} \quad f(y,n) := \sum_{v=1}^{|y|} y_v + \sum_{v'=1}^{|n|} n_{v'} \tag{19}$$

$$\text{s.t.} \quad l_{(i,j)} + y_v M_1 \geq v \qquad \forall v \in YES_{(i,j)} \tag{20}$$

$$l_{(i,j)} + n_{v'} M_2 \leq v' \qquad \forall v' \in NO_{(i,j)} \tag{21}$$

$$y_v \in \mathbb{B} \qquad \forall v \in YES_{(i,j)} \tag{22}$$

$$n_{v'} \in \mathbb{B} \qquad \forall v' \in NO_{(i,j)} \tag{23}$$

$$l_{(i,j)} \in \mathbb{R}_0^+ \tag{24}$$

Where $M1$ and $M2$ are constants who are 'big enough' and equations (25) & (26) assure that.

$$M_1 := -1 \times \min_{v} \{YES_{(i,j)}\} \tag{25}$$

$$M_2 := -1 \times \max_{v'} \{NO_{(i,j)}\} \tag{26}$$

By combining the three functions we can now construct a wish time for the event to happen. The function *time* then looks as follows.

$$\begin{aligned} time(i, x^{k+1}, d^{k+1}, t^{k+1}) := max\{&time_{\text{top}}(i, x^{k+1}), \\ &time_{\text{earliest}}(i, x^{k+1}, d^{k+1}, t^{k+1}), \\ &time_{\text{dm}}(i, x^{k+1}, d^{k+1})\} \end{aligned} \tag{27}$$

To summarize, we will give the full pseudo code in **Pseudo-code 1**.

Pseudo-code 1.

Input: time $t^{k+1} \in \mathbb{N}$, old disposition timetable x^k, a new delay state d^{k+1}, function $time(.,.,.,.)$
Output: new disposition timetable x^{k+1}

for $i \in \mathcal{E}$ **do**
    $x_i^{k+1} \leftarrow \infty$;
    if $x_i^k < t^{k+1}$ **then**
        $x_i^{k+1} \leftarrow x_i^k$
    end if
end for
while there is an event i with $x_i^{k+1} = \infty$ **do**
    $i \leftarrow$ choose an arbitrary element in $\{i \in \mathcal{E} | x_i^{k+1} = \infty\}$ with minimal $time(i, x^{k+1}, d^{k+1}, t^{k+1})$;
    $x_i^{k+1} \leftarrow time(i, x^{k+1}, d^{k+1}, t^{k+1})$
end while

---

4 Results

In this section we evaluate the online-strategies presented in section 3. In order to compare the results from us with Bauer and Schöbel [1] properly, we need to use the same methodology for the experiments.

Data. We were given a dataset that represents a subset of the network of the Dutch railway system. The *roll-out time* of this dataset is 8 hours. The roll-out time gives the length of the considered time horizon. The dataset does not contain headways as we refrain from headway activities. We use the delay generator DDA in order to simulate delays. To generate an instance d^{final} of DDA, we first need to know how many activities we want to be delayed. Just like in Bauer and Schöbel [1], we distinguish 3 cases: 10, 20 & 60 delayed (waiting and driving) activities. We then choose these activities randomly from the dataset and assign a random value u to each of these activities. The value u depends on the given delay scenario:

- Scenario WEAK: $u \sim$ UNI[1 min,3 min]
- Scenario MEDIUM: $u \sim$ UNI[3min, 15min]
- Scenario STRONG: $u \sim$ UNI[15min, 18min]

For the preprocessing phase of the AHRS strategy we always use 1000 simulation runs. In order to evaluate which strategy performs better, we generate 1000 instances of d^{final} and apply both strategies on the same instances.

Visualisation. All experiments are visualized as box-and-whiskers plots. The applied online strategies can be found on the x-axis. The y-axis gives the relative quality of the solutions by comparing it with the tight lower bound described in section 3.1. The value $y = 1$ means that the applied online strategy performed optimally as it is equal to solution of the corresponding offline problem. The value $y = 1.25$ means that the objective value of the online strategy is 25% larger than the objective of the offline solution. The bottom and top of the box give the 25th and the 75th percentiles. The central mark on the box is the median and the whiskers give the smallest and highest observations without outliers. An outlier is an observation outside 1.5 interquartile range of the upper quartile or lower quartile. We depict outliers as (red) plus signs.

Runtime. We implemented the experiments in JAVA using CPLEX as MIP-solver. They were executed on a Pentium Dual-Core CPU E5500 running Windows 7. It is clocked at 2.8 GHz and has 3GB of RAM. We did not measure the runtimes in great detail because of our different focus. Therefore, no runtimes are reported.

Results. The box-and-whiskers plots of our experiments are presented in Figure 2, where the titles refer to the format: DELAY SCENARIO/NUMBER OF ARCS DELAYED. We conducted the same experiments as in Bauer and Schöbel [1] to see if the results obtained by Bauer and Schöbel [1] still hold for our Dutch railway data.

Delay scenario WEAK. By looking at the top row of Figure 2 we observe the following. It is clear that in each WEAK scenario the OnlineMIP is being outperformed by the AHRS strategy. We observe fewer outliers for the AHRS. Also, for each WEAK scenario, the median lies significantly below the median of the OnlineMIP strategy. At last, the upper and lower quartile of the boxes obtained by AHRS are closer to each other. This implies less volatility for AHRS in contrast to the OnlineMIP, which means that AHRS is more stable and thus more reliable than the OnlineMIP. We conclude that for each WEAK scenario the OnlineMIP is being outperformed by the AHRS strategy.

Comparison. We first note that the figure presented in Bauer and Schöbel [1] where they present their results are very small. Therefore we cannot say much about outliers as these are hardly noticeable. Compared with Bauer and Schöbel [1] the results for the WEAK scenarios seem to match our results. In [1] the OnlineMIP strategy is more volatile and is being outperformed by the AHRS strategy, except for the WEAK/60 scenario where both strategies perform equally well.

Delay scenario MEDIUM. By looking at the second row of Figure 2 we observe the following. In case of 10 delayed activities the AHRS strategy gives better results. We see lesser volatility and the median of the AHRS strategy lies below the median of the OnlineMIP strategy. However, as the

amount of delayed activities increase we see that the OnlineMIP strategy is starting to give better results than the AHRS strategy, which is becoming more volatile. With 60 delayed activities the OnlineMIP strategy became less volatile in contrast to the case where only 10 activities were delayed and outperforms the AHRS strategy. The AHRS strategy with 60 delayed activities performs poorly as it is very volatile and produces a median higher than the median of the OnlineMIP strategy. With 20 delayed activities the AHRS strategy performs better in the sense that the median is lower. However, the amount of outliers is much higher compared to the OnlineMIP strategy. We suspect that there is a transition point between 20 arcs delayed and 60 arcs delayed from where the OnlineMIP strategy starts giving better results than the AHRS strategy. We conclude that for 10 and 20 delayed activities the AHRS strategy gives better results than the OnlineMIP. However, the OnlineMIP strategy gives better results with many (60) delayed activities.

Comparison. In contrast to our results, in Bauer and Schöbel [1] the AHRS strategy is being outperfomed in each MEDIUM delay scenario. In our case the AHRS gives worse results when increasing the number of delayed activities. In [1] both strategies perfom better when we increase the amount of delayed arcs, with the OnlineMIP constantly outperforming the AHRS strategy.

Delay scenario STRONG. By looking at the bottom row of Figure 2 we observe the following. We can see the same pattern as in the MEDIUM delay scenario. The AHRS performs better with 10 and 20 delayed arcs. However, the amount of outliers is much larger for the AHRS in these scenarios. While the amount of delayed activities increases, the OnlineMIP strategy is starting to give better results with a decrease in volatility and the median. The OnlineMIP seems very stable in the STRONG/60 scenario. The AHRS strategy however, seems to be more volatile when the number of delayed arcs increases. We conclude that the AHRS performs good with 10 and 20 delayed activities, however it is being outperformed by the OnlineMIP strategy when increasing the amount of delayed activities (60).

Comparison. The results do not match our findings completely. We see again that in Bauer and Schöbel [1] both strategies perform better when the amount of delayed arcs increases. However, in our case the volatility of the AHRS strategy increases when we increase the amount of delayed activities.

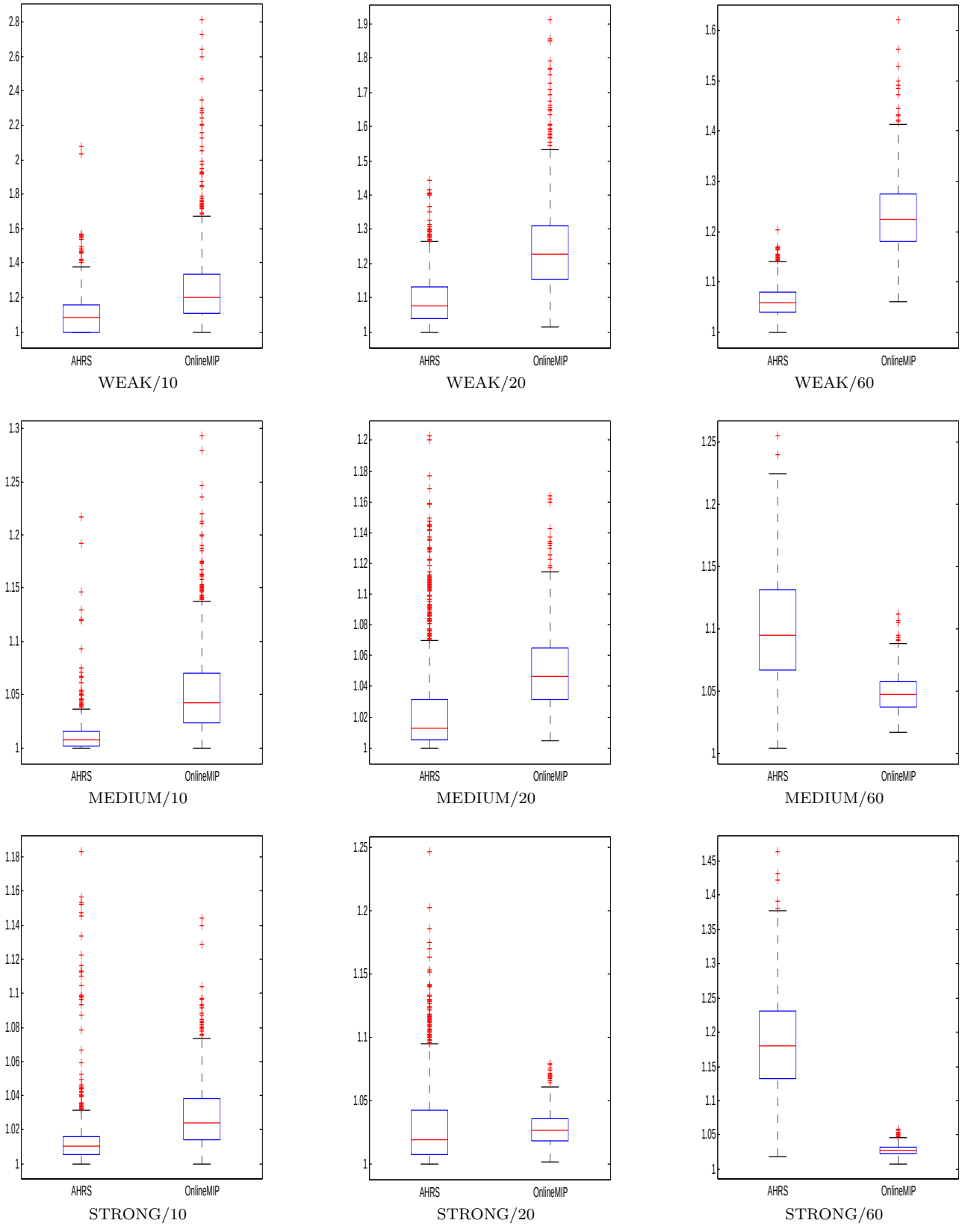


Figure 2: Results of our experiments with 1000 simulations.

5 Conclusion

Our goal of this report was to check whether the results obtained in [1] still hold for our Dutch railway data. To do this, we have introduced the *delay management problem*. Two strategies were given in order to solve the online version of the problem, namely the *Ad-Hoc Re-Scheduling* strategy which uses heuristic functions, and the *OnlineMIP* strategy which make use of the *OfflineILP* proposed in [1]. We evaluated both strategies by comparing the results obtained with Dutch railway data and the data used in Bauer and Schöbel [1]. We did not see entirely the same results as in [1]. For the WEAK delay scenarios we see many similarities. For the MEDIUM and STRONG scenarios however, we see slightly different patterns. The AHRS strategy seems to perform better than the OnlineMIP in case when not many activities are delayed (10, 20). However, when we increase the amount of delayed activities the AHRS strategy becomes more volatile. In case of 60 delayed arcs the OnlineMIP performs better just like in [1]. We have not looked at the data used in [1] due to time limitations. The differences might be due to different characteristics of the data. It would be of great interest to know if this is the case. For this, further research is required.

References

[1] Reinhard Bauer and Anita Schöbel. Rules of thumb: practical online-strategies for delay management. *Public Transport*, 6(1-2):85–105, 2014. ISSN 1866-749X. doi: 10.1007/s12469-013-0082-8. URL `http://dx.doi.org/10.1007/s12469-013-0082-8`.

[2] Michael Schachtebeck and Anita Schöbel. To wait or not to wait—and who goes first? delay management with priority decisions. *Transportation Science*, 44(3):307–321, August 2010. ISSN 1526-5447. doi: 10.1287/trsc.1100.0318. URL `http://dx.doi.org/10.1287/trsc.1100.0318`.