Control of Delay Propagation in Railway Networks Using Max-Plus Algebra

Maxime Hoekstra

TU Delft

Control of Delay Propagation in Railway Networks Using Max-Plus Algebra

Report on behalf of the
Delft Institute of Applied Mathematics
as part of obtaining

the degree of

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

MAXIME HOEKSTRA

Delft University of Technology
Faculty of Electrical Engineering, Applied Mathematics and Computer Science
Delft Institute of Applied Mathematics

Delft, The Netherlands
November 2020

Supervisor
Dr. J.W. van der Woude

Other committee members
Dr. B. van den Dries

Abstract

In this thesis max-plus algebra is introduced and applied to the problem of controlling train delays. Two control strategies for the propagation of delays are discussed. The first is by letting certain trains run faster when a delay is detected, the second is by breaking connections between trains that have to wait for each other in order to enable passengers to changeover from one train to the other. The goal is to understand how to model the propagation of delays when different control strategies are applied, in order to provide train operators tools for making quick decisions on how to intervene when a delay is detected.

The models provided in the report are in the form of max-plus-linear systems and switching max-plus linear systems. These can be programmed in Python to automate the decision making. The report starts with providing a basic understanding of max-plus algebra, where also max-plus linear systems and switching max-plus linear systems are explained. Subsequently, a railway network is designed that serves as an example during this thesis. This railway network is modelled into a max-plus linear system and, additionally, a desirable train timetable for passengers is designed for this network by means of the power algorithm. The main results are two switching max-plus linear systems that model the propagation of delays when the two control strategies are applied. The report ends with a larger railway network at which all acquired knowledge is applied.

It can be concluded that the models in this thesis provide methods to calculate exactly how the delay propagates through the network when certain control strategies are applied and, based on that, decisions can be made quicker. Moreover, it is possible to calculate the consecutive departure times. As a result the passengers can be informed quickly about the new departure times as a consequence of the delay and how long it will take for the trains to run according to timetable again. This thesis adds the modelling of faster running trains to existing literature. We have seen that speeding up trains is also a control strategy to solve delays and can be modelled systematically.

Preface

This report is written for the purpose of obtaining the degree of Bachelor of Science in Applied Mathematics at the Delft University of Technology. In this thesis the conventional linear algebra that students learn in their bachelor is extended to max-plus algebra and applied to a practical problem: modelling train delays. For ten weeks I have worked on this problem. I enjoyed delving into new theory and working on providing methods to solve a very practical problem which many people benefit from. I would like to thank my supervisor Dr. J. W. van der Woude for the great supervision and support during this project by means of weekly meetings.

M. F. Hoekstra
Delft, October 2020

Contents

1 Introduction

Train delays are always inconvenient and cost a lot of money, time and annoyance. We want to solve them as quick as possible to prevent the delay from propagating through the whole railway network, causing even more hindrance. To be able to solve the delay quickly, it would be helpful if train operators had tools for deciding how to intervene when a delay is detected, since currently it happens somewhat intuitively. Train operators need to make a lot of tradeoffs when it comes to deciding which strategy to use to tackle the delay. That is why it would be useful if they have models to automate the decision making and to be able to visualize the consequences by simulating the effects of a control strategy. Therefore, the goal of this thesis is to model the propagation of delays when certain control strategies are applied.

The control strategies discussed in this thesis are 1. letting certain trains run faster when a delay is detected and 2. breaking connections between trains that have to wait for each other before they can depart. The first strategy of letting trains run faster adds to the existing literature about modelling railway networks. The models are constructed with max-plus algebra, which differs slightly from the conventional algebra in which we use addition and multiplication, but turns out to be more practical when modelling train departures.

This report starts with providing basic knowledge about max-plus algebra, max-plus-linear systems and switching max-plus-linear systems. In chapter 3 a simple railway network is constructed and modelled into a max-plus-linear system. Additionally, a desirable train timetable for the passengers is designed. In chapter 4 delays in the railway network are modelled and the two control strategies are applied. This leads to switching max-plus-linear models for the control strategies. In chapter 5 all required knowledge is applied to a larger, more realistic railway network. The report ends with a conclusion (chapter 6) and some remarks that may lead to further research (chapter 7).

2 Max-plus algebra

A railway network is an example of a discrete event system (DES). A DES deals with sequences of events. In this case the events are trains leaving the station but other examples are papers being printed and serial production systems. These events need to be modelled and timed, subject to synchronization constraints. For example, a train may not depart before another train has arrived [Heidergott et al., 2006]. The modelling of discrete event systems traditionally leads to nonlinear systems in conventional algebra. However, there are certain types of DESs that are described linearly in max-plus algebra. These DESs are called max-plus-linear (MPL) systems [Van den Boom and De Schutter, 2012]. This chapter begins with providing a basic understanding of max-plus algebra (2.1). Thereafter some extensions of this theory, that are used during this thesis, are explained. These are max-plus-linear (MPL) systems (2.2) and switching max-plus-linear (SMPL) systems (2.3).

2.1 Basic concepts and definitions

In max-plus algebra the addition ($+$) and multiplication ($\times$) in conventional algebra are replaced with maximization ($\oplus$) and addition ($\otimes$), respectively. Maximization is used for the synchronization of events; an event can only start as soon as all connected events are finished. The addition operation follows from the time instants at which events occur; the time an event is finished equals the time at which it started plus the time it takes to complete its task [Kersbergen, 2015]. These are the only operations being used in max-plus algebra and they lead to linear systems that work analogously to conventional linear algebra. In this section an overview of the basics of max-plus algebra is given, based on the theory in the book "Max Plus at Work" [Heidergott et al., 2006].

2.1.1 Basic definitions

We define two important constants in max-plus algebra:

$$\varepsilon \coloneqq -\infty, \qquad e \coloneqq 0. \tag{2.1}$$

Let $\mathbb{R}_{\max}$ be the set $\mathbb{R} \cup \{\varepsilon\}$. Let $a, b \in \mathbb{R}_{\max}$, then the operations $\oplus$ and $\otimes$ (pronounced "o-plus" and "o-times") are defined by

$$a \oplus b \coloneqq \max(a, b), \qquad a \otimes b \coloneqq a + b. \tag{2.2}$$

For example, $5 \oplus 3 = \max(5, 3) = 5$ and $5 \otimes 3 = 5 + 3 = 8$.

We now define *max-plus algebra* as the set $\mathbb{R}_{\max}$ together with the operations $\oplus$ and $\otimes$, and denote it by

$$\mathcal{R}_{\max} = (\mathbb{R}_{\max}, \oplus, \otimes, \varepsilon, e).$$

As in conventional algebra, we let the operation $\otimes$ have priority over the operation $\oplus$. So $5 \otimes -9 \oplus 7 \otimes 1$ means $(5 \otimes -9) \oplus (7 \otimes 1)$ and is equal to $\max(5 - 9, 7 + 1) = 8$. A list of algebraic properties of max-plus algebra is given on the next page.

- Associativity:
$$\forall a, b, c \in \mathbb{R}_{\max} : \ a \oplus (b \oplus c) = (a \oplus b) \oplus c$$
and
$$\forall a, b, c \in \mathbb{R}_{\max} : \ a \otimes (b \otimes c) = (a \otimes b) \otimes c.$$

- Communitativity:
$$\forall a, b \in \mathbb{R}_{\max} : \ a \oplus b = b \oplus a \quad \text{and} \quad a \otimes b = b \otimes a.$$

- Distributivity of $\otimes$ over $\oplus$:
$$\forall a, b, c \in \mathbb{R}_{\max} : \ a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c).$$

- Existence of a zero element:
$$\forall a \in \mathbb{R}_{\max} : \ a \oplus \varepsilon = \varepsilon \oplus a = a.$$

- Existence of a unit element:
$$\forall a \in \mathbb{R}_{\max} : \ a \otimes e = e \otimes a = a.$$

- The zero is absorbing for $\otimes$:
$$\forall a \in \mathbb{R}_{\max} : \ a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon.$$

- Idempotency of $\oplus$:
$$\forall a \in \mathbb{R}_{\max} : \ a \oplus a = a.$$

2.1.2 Vectors and matrices

We can extend the theory to vectors and matrices. Let $\mathbb{R}_{\max}^{n \times m}$ be the set of $n \times m$ matrices with underlying max-plus algebra. For $n \in \mathbb{N}$ with $n \neq 0$, define $\underline{n} \coloneqq \{1, 2, \ldots, n\}$. Then for $A, B \in \mathbb{R}_{\max}^{n \times m}$, the *matrix addition* $A \oplus B$ is defined by

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \tag{2.3}$$

for $i \in \underline{n}$ and $j \in \underline{m}$.

For $A \in \mathbb{R}_{\max}^{n \times l}$ and $B \in \mathbb{R}_{\max}^{l \times m}$, the *matrix multiplication* $A \otimes B$ is defined by

$$[A \otimes B]_{ik} = \bigoplus_{j=1}^{l} a_{ij} \otimes b_{jk} = \max_{j \in \underline{l}} \{a_{ij} + b_{jk}\} \tag{2.4}$$

for $i \in \underline{n}$ and $k \in \underline{m}$.

At last, for $A \in \mathbb{R}_{\max}^{n\times m}$ and $\alpha \in \mathbb{R}_{\max}$, the *scalar multiple* $\alpha \otimes A$ is defined by

$$[\alpha \otimes A]_{ij} = \alpha \otimes a_{ij} \tag{2.5}$$

for $i \in \underline{n}$ and $j \in \underline{m}$.

For example, let $A = \begin{pmatrix} e & \varepsilon \\ 3 & 2 \end{pmatrix}$ and $B = \begin{pmatrix} -1 & 11 \\ 1 & \varepsilon \end{pmatrix}$. Then

$$A \oplus B = \begin{pmatrix} \max(e,-1) & \max(\varepsilon,11) \\ \max(3,1) & \max(2,\varepsilon) \end{pmatrix} = \begin{pmatrix} e & 11 \\ 3 & 2 \end{pmatrix}$$

and

$$A \otimes B = \begin{pmatrix} -1 & 11 \\ 3 & 14 \end{pmatrix},$$

because according to (2.4), $[A \otimes B]_{11} = e \otimes (-1) \oplus \varepsilon \otimes 1 = \max(0-1, -\infty+1) = -1$, and the other elements are calculated similarly.

We can also define *matrix powers*. For $A \in \mathbb{R}_{\max}^{m\times n}$, denote the kth power of A by

$$A^{\otimes\, k} \coloneqq \underbrace{A \otimes A \otimes \cdots \otimes A}_{k \text{ times}} \tag{2.6}$$

for $k \in \mathbb{N}$ with $k \neq 0$ and $A^{\otimes\, 0} \coloneqq E(n,n)$, with

$$[E(n,m)]_{ij} \coloneqq \begin{cases} e \text{ for } i = j \\ \varepsilon \text{ otherwise} \end{cases} .$$

We end this subsection about matrices with eigenvalues and eigenvectors. Let $A \in \mathbb{R}_{\max}^{n\times n}$ be a square matrix. If $\lambda \in \mathbb{R}_{\max}$ is a scalar and $v \in \mathbb{R}_{\max}$ is a vector that contains at least one finite element such that

$$A \otimes v = \lambda \otimes v, \tag{2.7}$$

then λ is called an *eigenvalue* of A and v an *eigenvector* of A associated with eigenvalue λ. A square matrix can have more than one eigenvalue. The eigenvectors are also not unique. Suppose v is an eigenvector, then $\alpha \otimes v$, with α an arbitrary finite number, is also an eigenvector.

2.2 Max-plus-linear systems

As mentioned before, max-plus linear (MPL) systems are discrete event systems that result in linear models in max-plus algebra. A characteristic of MPL systems is that synchronization occurs, but there is no concurrency and no choice. No concurrency in our case means that no more than one train can occupy a track. No choice means that a train follows a fixed sequence of stations. However, there is synchronization, meaning that

trains can only depart as soon as other trains are arrived, in order to enable passenger changeovers. MPL systems are modelled in the following form:

$$x(k) = A \otimes x(k-1) \oplus B \otimes u(k) \tag{2.8}$$
$$y(k) = C \otimes x(k), \tag{2.9}$$

with $A \in \mathbb{R}_{\max}^{n \times n}, B \in \mathbb{R}_{\max}^{n \times m}$ and $C \in \mathbb{R}_{\max}^{o \times n}$. Here, n is the number of states, m is the number of inputs, o is the number of outputs and k is the event counter [Kalamboukis, 2018]. $x(k)$ describes the time instants at which the internal events occur for the kth time, $u(k)$ describes the time instants at which the input events occur for the kth time and $y(k)$ describes the time instants at which the output events occur for the kth time [Van den Boom and De Schutter, 2012].

In the next chapters it becomes clear how to set up a MPL model for a simple railway network. In this case the vector $x(k)$ describes the kth departure times of the trains. The input $u(k)$ will be the scheduled departure times of the trains at event step k (they are predefined in a timetable). The output equation (2.9) will be excluded from this thesis, since the output of the system $y(k)$ equals the state vector $x(k)$.

2.3 Switching max-plus-linear systems

When we are solely using a MPL system, the structure of the model is fixed. However, in this thesis the propagation of delays and ways to control this propagation are researched. Therefore we need to be able to model changes in the structure of the system. That is why switching max-plus-linear (SMPL) systems are introduced. SMPL systems are MPL systems where switching is allowed between different modes of operation. Each mode represents a different state of the model, in which for instance the synchronization is changed or a certain train will drive faster. This results in different system matrices for each mode. SMPL systems are modelled in the following form:

$$x(k) = A^{l(k)} \otimes x(k-1) \oplus B^{l(k)} \otimes u(k) \tag{2.10}$$
$$y(k) = C^{l(k)} \otimes x(k), \tag{2.11}$$

in which $l(k) \in \{1, 2, \dots n_L\}$ represents the mode that is valid at event k, with n_L the number of possible modes. $A^{l(k)} \in \mathbb{R}_{\max}^{n \times n}, B^{l(k)} \in \mathbb{R}_{\max}^{n \times m}$ and $C^{l(k)} \in \mathbb{R}_{\max}^{o \times n}$ are the system matrices for mode l.

For each event k, the mode of the system $l(k)$ is determined by a switching mechanism $z(k)$. This switching variable may depend on the previous state $x(k-1)$, the previous mode $l(k-1)$, the input $u(k)$ and a control variable $v(k)$:

$$z(k) = \phi(x(k-1), l(k-1), u(k), v(k)) \in \mathbb{R}_{\max}^{n_z}. \tag{2.12}$$

The $\mathbb{R}_{\max}^{n_z}$ is partitioned into n_L subsets $\mathcal{Z}^i$, with $i = 1, 2, \dots n_L$. The mode $l(k)$ is now determined by the $\mathcal{Z}^i$ the $z(k)$ is in. When $z(k) \in \mathcal{Z}^i$ the system switches to mode i [Van den Muijsenberg, 2015].

3 Modelling of a simple railway network

For the analysis of delays in a railway network we design a simple fictional railway network, which will serve as an example during this report. At this example the theory in chapter 2 is applied. The network is a modification of a railway network used in the book of [Heidergott et al., 2006]. In this chapter the form of the railway network is explained (3.1), the network is modelled into a MPL system (3.2) and a desirable train timetable for the network is designed (3.3).

3.1 The railway network

Consider a railway network with two stations, S_1 and S_2, as indicated in Figure 1. The stations are connected by two tracks. One track runs from S_1 to S_2 and we assume the travel time on this track to be 5 time units. The other track runs from S_2 to S_1 and has a travel time of 9. Together, these two tracks form a circuit. The train on the track from S_1 to S_2 continues on the track from S_2 to S_1. Our railway network contains two other circuits, which connect the suburbs of a city to its main station. The left circuit has a travel time of 3, the right circuit has a travel time of 5 time units. Each of the four tracks is occupied by a train.

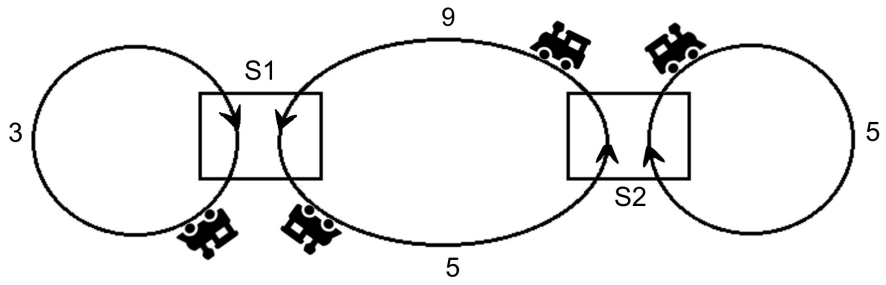

Figure 1: A simple fictional railway network.

The trains arriving at each station should wait for each other in order to allow the changeover of passengers. The transfer time for passengers is set to 2 time units. This means that as soon as both trains have arrived at a station, they will wait 2 time units in order for passengers to changeover from one train to the other.

3.2 The max-plus-linear model

In this section the previously described railway network is modelled into a MPL system as described in section 2.2. The model should meet five criteria [Heidergott et al., 2006]:

1. The travel times of the trains along each of the tracks is fixed and given.

2. The frequency of the trains (the number of departures per unit of time) must be as high as possible.

3. The frequency of the trains must be the same along all four tracks, yielding a timetable with regular departure times.

4. The trains arriving at a station should wait for each other in order to allow the changeover of passengers.

5. The trains at a station depart the station as soon as they are allowed.

Let $x_i(k)$ denote the kth departure time of the train going in direction i, with $i = 1, 2, 3, 4$. These departure times are the events in the model. So $x_3(0)$ is the first train going in direction 3, which is the track running from S1 to S2. In Figure 2 the tracks in the different directions $i = 1, 2, 3, 4$ are labeled in red.

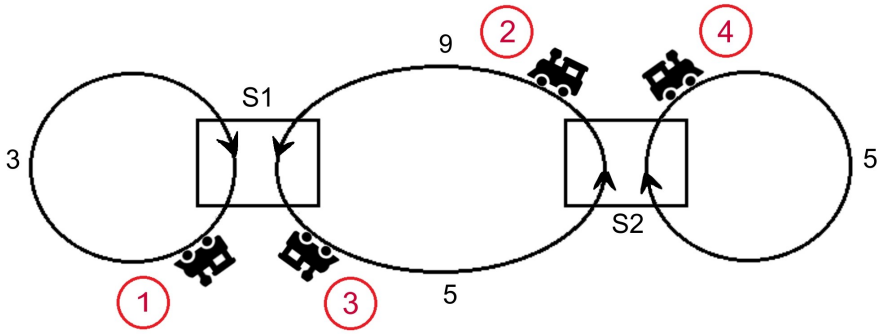

Figure 2: A simple fictional railway network with the different directions labeled in red.

Let a_{ij} denote the travel time of the train going in direction j, that is connected to the train going in direction i. Let t be the transfer time to enable passengers to changeover. From the rules given in the previous subsection, it follows that for x_3 (the train traveling in the direction 3):

$$x_3(k) \geq a_{32} + x_2(k-1) + t, \tag{3.1}$$
$$x_3(k) \geq a_{31} + x_1(k-1) + t. \tag{3.2}$$

Equation (3.1) follows from the assumption that the train going in direction 2 continues in direction 3. Hence the train in direction 3 can depart as soon as the train in direction 2 has arrived, which is at $a_{32} + x_2(k-1)$, plus the transfer time, which is t. Equation (3.2) follows from the fourth rule in the previous section: the train going in direction 3 should wait until the train going in direction 1 has arrived, so that passengers can changeover at station 1. Since it is more convenient to include the transfer time t in the travel time a_{ij}, we leave this constant out of the equation. Then for the departure time of the train in direction 3 it holds that:

$$\begin{aligned} x_3(k) &\geq \max(a_{32} + x_2(k-1), a_{31} + x_1(k-1)) \\ &= a_{32} \otimes x_2(k-1) \oplus a_{31} \otimes x_1(k-1) \\ &= 11 \otimes x_2(k-1) \oplus 5 \otimes x_1(k-1), \end{aligned} \tag{3.3}$$

where $a_{32} = travel\ time\ +\ transfer\ time = 9 + 2 = 11$ and $a_{31} = travel\ time\ + transfer\ time = 3 + 2 = 5$. Since the frequency of the trains should be as high as possible according to the second criterion, we replace the $\geq$-sign in (3.3) with an $=$-sign.

The departure times of the trains in direction 1, 2 and 4 are described similarly. This leaves us:

$$\begin{aligned} x_1(k) &= 5 \otimes x_1(k-1) \oplus 11 \otimes x_2(k-1), \\ x_2(k) &= 7 \otimes x_3(k-1) \oplus 7 \otimes x_4(k-1), \\ x_3(k) &= 5 \otimes x_1(k-1) \oplus 11 \otimes x_2(k-1), \\ x_4(k) &= 7 \otimes x_3(k-1) \oplus 7 \otimes x_4(k-1). \end{aligned} \tag{3.4}$$

In general,

$$x_i(k) = \bigoplus_{j=1}^{n} a_{ij} \otimes x_j(k-1) \tag{3.5}$$

for $i = 1, 2, \ldots n$, with n the number of trains. We set $a_{ij} = \varepsilon$ when there is no connection between train i and train j. These values will not participate in the maximum operation. Remark that in this example the departure times of the train in direction 1 are the same as the ones of the train in direction 3, likewise for train 2 and train 4.

The equations in (3.4) can be written in matrix form as:

$$x(k) = A \otimes x(k-1), \tag{3.6}$$

with

$$A = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix}.$$

This matrix is larger than the system matrix in the example in the book of [Heidergott et al., 2006], which is a 2×2 matrix. That is because Heidergott et al. (2006) defined x_i as the departure time of the two trains at station i, hence the vector $x(k)$ contains an element for each of the two stations. In this thesis we have chosen to define x_i as the departure time of the train going in direction i, causing the vector $x(k)$ to have an element for each of the four directions. This will be more convenient for the analysis of breaking connections between trains, later in this thesis. Breaking connections causes the two trains at a station to have different departure times, because they are not obligated to wait for each other anymore. Therefore it is clearer to assign a departure time to the trains in each direction separately.

There is one last criterion that is included in the MPL model, that is the fifth rule "The trains at a station depart the station as soon as they are allowed". This also means that the trains may not leave before the scheduled departure times in the train timetable, otherwise this would be confusing for the passengers. Hence when $d(k)$ denotes the kth

scheduled departure times in the train timetable, it must hold that

$$x(k) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k). \tag{3.7}$$

The scheduled departure times $d(k)$ are determined beforehand by means of the power algorithm. This will be explained in the next section.

We have now arrived at a MPL system in the form of (2.8), where $x(k)$ are the kth departure times, A is the state matrix of which the elements represent the travel time plus transfer time, and $d(k)$ are the scheduled departure times which serve as input of the system. As mentioned before, the output equation (2.9) is omitted since the output $y(k)$ equals the state $x(k)$.

3.3 Power algorithm: designing a desirable train timetable

In this section the train timetable is designed such that the frequency of the trains is as high as possible and the timetable is convenient for passengers. Let $d(k)$ denote the kth departure times in the train timetable. We are trying to find the best first departure times of the day, $d(0)$, that lead to a regular train timetable. The timetable is regular when the time between two consecutive departures in the same direction, the interdeparture time, is constant. For this the power algorithm is applied.

With the power algorithm the eigenvalue and eigenvector of a $n \times n$ matrix can be computed. It turns out that the eigenvector of matrix A is a suitable choice for $d(0)$ and the eigenvalue will then be the minimal interdeparture time. The *power algorithm* is given below [Heidergott et al., 2006].

1. Take an arbitrary initial vector $x(0)$ such that $x(0)$ has at least one finite element.
2. Compute $x(k) = A \otimes x(k-1)$ until there are integers p, q with $p > q \geq 0$ and a real number c, such that $x(p) = x(q) \otimes c$, i.e., until a periodic regime is reached.
3. Compute as the eigenvalue $\lambda = c/(p-q)$.
4. Compue as an eigenvector $v = \bigoplus_{j=1}^{p-q} (\lambda^{\otimes(p-q-j)} \otimes x(q+j-1))$.

In appendix A the power algorithm is programmed in Python. For our example we

start with $x(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ in step 1. Next, step 2 gives us

$$\begin{aligned} x(1) &= \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 11 \\ 7 \\ 11 \\ 7 \end{pmatrix}, \\ x(2) &= \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 11 \\ 7 \\ 11 \\ 7 \end{pmatrix} = \begin{pmatrix} 18 \\ 18 \\ 18 \\ 18 \end{pmatrix}. \end{aligned} \tag{3.8}$$

Hence $x(2) = x(0) \otimes 18$ and so $p = 2$, $q = 0$ and $c = 18$. Subsequently the eigenvalue can be calculated as in step 3. We get $\lambda = 18/(2-0) = 9$. This means that every 9 time units the trains will depart in every direction. This is the smallest interdeparture time possible. However, for the passengers it is easier if the trains depart every 10 time units since it is easier to remember and to deal with, i.e.,

$$d(k) = 10 \otimes d(k-1). \tag{3.9}$$

The only thing left to know for the train timetable is a suitable starting vector with departure times, $d(0)$. When there are no delays, we must have

$$A \otimes x(k-1) \leq d(k), \tag{3.10}$$

so that the trains run according to the timetable. Substituting (3.9) into (3.10) gives us

$$A \otimes x(k-1) \leq 10 \otimes d(k-1) \tag{3.11}$$

and thus also

$$A \otimes x(0) \leq 10 \otimes d(0). \tag{3.12}$$

When there are no initial delays, we have $x(0) = d(0)$ and hence it must hold that

$$A \otimes d(0) \leq 10 \otimes d(0) = d(1). \tag{3.13}$$

For the eigenvector v of A, it holds that $A \otimes v = 9 \otimes v \leq 10 \otimes v$. Therefore, choosing the eigenvector for $d(0)$ satisfies (3.13). Step 4 in the power algorithm gives us

$$\begin{aligned} v &= 9^{\otimes 1} \otimes x(0) \oplus 9^{\otimes 0} \otimes x(1) \\ &= \begin{pmatrix} 9 \\ 9 \\ 9 \\ 9 \end{pmatrix} \oplus \begin{pmatrix} 11 \\ 7 \\ 11 \\ 7 \end{pmatrix} = \begin{pmatrix} 11 \\ 9 \\ 11 \\ 9 \end{pmatrix}. \end{aligned} \tag{3.14}$$

As mentioned in subsection 2.1.2, the eigenvector is unique up to an additive constant, hence $\begin{pmatrix} 2 \\ 0 \\ 2 \\ 0 \end{pmatrix}$ is also an eigenvector.

Note that the eigenvector of A is not the only vector that satisfies (3.13). For instance, the reader can check that $\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ is also a suitable choice for $d(0)$, that satisfies (3.13).

We have now arrived at a desirable train timetable for passengers. The $d(k)$ are listed below

$$d(0) = \begin{pmatrix} 2 \\ 0 \\ 2 \\ 0 \end{pmatrix}, \; d(1) = \begin{pmatrix} 12 \\ 10 \\ 12 \\ 10 \end{pmatrix}, \; d(2) = \begin{pmatrix} 22 \\ 20 \\ 22 \\ 20 \end{pmatrix}, \; d(3) = \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix}, \; \ldots \tag{3.15}$$

4 Control strategies for the propagation of delays

In this chapter methods to solve a train delay will be introduced. To test the effect of these methods, we first need to know how to model the propagation of delays when there is no intervention. This will be discussed in (4.1). Thereafter two control strategies will be discussed. One is to allow certain trains to run faster (4.2) and the second method is breaking connections, whereby the passenger changeover constraint is neglected (4.3). Both control strategies are modelled in a switching max-plus-linear model to automate the system.

4.1 Modelling the propagation of delays

We speak of a delay when for a certain train j, $x_j(k) > d_j(k)$. To find out the subsequent delayed departure times, we apply the MPL model (3.7) until $x_i(k) = d_i(k)$ for all i. This will eventually happen because the network contains some slack time. Slack time appears when the train is ready to depart before the scheduled departure time indicated in the timetable. Since the trains are not allowed to depart before the scheduled departure time, the slack time serves as some buffer time to absorb delays. For example, the trains are ready to leave the second time at

$$\begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 11 \\ 9 \\ 11 \\ 9 \end{pmatrix},$$

while the second scheduled departure times in the timetable are $d(1) = \begin{pmatrix} 12 \\ 10 \\ 12 \\ 10 \end{pmatrix}$. The "$\oplus\, d(k)$" part in the MPL model (3.7) makes sure that the trains do not leave before $d(1)$.

Let's have a look at an example of the propagation of a delay. Suppose at $k = 1$ the train in direction 2 has a delay of 8. Hence we have $x(1) = \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix}$. Applying the MPL model (3.7) gives

$$x(2) = \begin{pmatrix} 29 \\ 20 \\ 29 \\ 20 \end{pmatrix},\ x(3) = \begin{pmatrix} 34 \\ 36 \\ 34 \\ 36 \end{pmatrix},\ \ldots,\ x(8) = \begin{pmatrix} 83 \\ 80 \\ 83 \\ 80 \end{pmatrix},\ x(9) = \begin{pmatrix} 92 \\ 90 \\ 92 \\ 90 \end{pmatrix}. \tag{4.1}$$

Hence at $k = 9$, $x(9) = d(9)$ and all trains will leave on time again. Note that the elements of x are never smaller than the elements of d, so that the timetable is always leading.

The MPL model can be programmed in Python such that the $x(k)$ are calculated, see appendix A.

If we want to know the magnitude of the total delay in the system and which trains are delayed, the vectors $z(k) = x(k) - d(k)$ are calculated. This vector displays the magnitude of the delay of each train at event k. Note that always $z_i(k) \geq 0$ for all i. We get

$$z(1) = \begin{pmatrix} 0 \\ 8 \\ 0 \\ 0 \end{pmatrix}, \; z(2) = \begin{pmatrix} 7 \\ 0 \\ 7 \\ 0 \end{pmatrix}, \; z(3) = \begin{pmatrix} 2 \\ 6 \\ 2 \\ 6 \end{pmatrix}, \; \ldots, \; z(8) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{4.2}$$

and the total delay in the system is 72 (sum all elements of all z), where the first delay of 8 is not included since that delay already happened and cannot be influenced anymore.

4.2 Control strategy: Let trains run faster

One way to catch up a train delay is by letting trains run faster for a period of time until the trains can run according to the timetable again. In general, not every train is allowed to run faster because of safety reasons. But when a train travels a long distance through grasslands for example, it is allowed to slightly speed up the train. In our fictional railway network we assume that only the train in direction 2 is allowed to reduce its travel time to 7 instead of 9. In section 2.3 was mentioned that the structure of a MPL system is fixed. This means that when we want to reduce the travel time of one train, this leads to a change in the state matrix A, and consequently we need to build a switching max-plus-linear (SMPL) model.

4.2.1 Switching max-plus-linear model

The SMPL model consists of two modes: one contains the state matrix from the original MPL model, the second mode contains the altered state matrix where the train in direction 2 runs faster. The model should constantly switch between the two modes in such a way that the trains run as much as possible according to timetable. The SMPL model is described as in section 2.3.

Let $l(k) \in \{1, 2\}$. Recall that $l(k)$ represents the mode that is valid at event k. If $l(k) = 1$, the vector $x(k)$ is calculated with matrix 1 (the original matrix). If $l(k) = 2$, the vector $x(k)$ is calculated with matrix 2 (the altered matrix). Note that the matrix of mode 2 has a smaller eigenvalue and consequently a smaller interdeparture time. However, since we always take the maximum with $d(k)$, the train timetable is always leading and the trains will therefore never leave before the scheduled departure times. This is more convenient for the passengers.

Mode 1

$$x(k) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k). \tag{4.3}$$

Mode 2

$$x(k) = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k). \tag{4.4}$$

For each event k, the mode of the system $l(k)$ is determined by the switching variable z, that depends on the previous state $x(k-1)$ and the input variable $d(k)$. We define

$$z(k) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k) \in \mathbb{R}^4_{\max}. \tag{4.5}$$

The space $\mathbb{R}^4_{\max}$ is partitioned into two subsets, one for each mode:

$$\mathcal{Z}^1 = \{z(k) |\ z(k) = d(k)\}, \tag{4.6}$$
$$\mathcal{Z}^2 = \{z(k) |\ \exists\ i \in \{1,2,3,4\} :\ z_i(k) > d_i(k)\}. \tag{4.7}$$

If $z(k) \in \mathcal{Z}^1$, the system switches to mode 1 and if $z(k) \in \mathcal{Z}^2$, the system switches to mode 2. Both modes are calculated with the same Python program but with a different state matrix, see appendix A. Note that $z(k)$ is equal to the original MPL system for the train departures. The original matrix is used for $z(k)$ because the switching variable should test whether it is possible to let the trains run as usual. When travelling with the original travel times leads to a delay, the system should switch to the altered matrix. Conversely, suppose the altered matrix is used in the switching variable, then the outcome of $z(k)$ may be that $z(k) = d(k)$, so that the system switches to the original matrix to calculate $x(k)$. However, when subsequently the original matrix is used to calculate $x(k)$, it is possible that there is still a delay present. For example, suppose at $k = 3$ we have that

$$z(3) = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 21 \\ 22 \\ 21 \\ 22 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} \in \mathcal{Z}^1,$$

so that the system switches to mode 1 to calculate $x(3)$. But then

$$x(3) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 21 \\ 22 \\ 21 \\ 22 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 33 \\ 30 \\ 33 \\ 30 \end{pmatrix} \text{ instead of } \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix},$$

hence the system switched to early. That is why the original matrix is used in the definition of $z(k)$.

4.2.2 Example of a delay

Suppose at $k = 1$ the train in direction 2 has a delay of 8. In section 4.1 the propagation of this delay is calculated. In this example the SMPL model is applied. We have:

$$x(1) = \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix}$$

$$z(2) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 22 \\ 20 \\ 22 \\ 20 \end{pmatrix} = \begin{pmatrix} 29 \\ 20 \\ 29 \\ 20 \end{pmatrix} \in \mathcal{Z}^2,$$

hence l(2)=2, so the system switches to mode 2 to calculate x(2).

$$x(2) = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 22 \\ 20 \\ 22 \\ 20 \end{pmatrix} = \begin{pmatrix} 27 \\ 20 \\ 27 \\ 20 \end{pmatrix}$$

$$z(3) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 27 \\ 20 \\ 27 \\ 20 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 32 \\ 34 \\ 32 \\ 34 \end{pmatrix} \in \mathcal{Z}^2$$

$$x(3) = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 27 \\ 20 \\ 27 \\ 20 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 32 \\ 34 \\ 32 \\ 34 \end{pmatrix}$$

$$z(4) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 32 \\ 34 \\ 32 \\ 34 \end{pmatrix} \oplus \begin{pmatrix} 42 \\ 40 \\ 42 \\ 40 \end{pmatrix} = \begin{pmatrix} 45 \\ 41 \\ 45 \\ 41 \end{pmatrix} \in \mathcal{Z}^2$$

$$x(4) = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 32 \\ 34 \\ 32 \\ 34 \end{pmatrix} \oplus \begin{pmatrix} 42 \\ 40 \\ 42 \\ 40 \end{pmatrix} = \begin{pmatrix} 43 \\ 41 \\ 43 \\ 41 \end{pmatrix}$$

$$z(5) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 43 \\ 41 \\ 43 \\ 41 \end{pmatrix} \oplus \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} = \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} \in \mathcal{Z}^1,$$

hence l(5)=1 and the system switches back to mode 1.

$$x(5) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 43 \\ 41 \\ 43 \\ 41 \end{pmatrix} \oplus \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} = \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix}.$$

By letting the train in direction 2 run faster, the delay is solved at $k = 5$, compared to $k = 9$ when no control is applied. Next to that the total delay in the system is now 22 compared to 72. The switching variable $z(k)$ decides if $x(k)$ will be calculated in mode 1 or mode 2.

The steps can be interpreted as follows. To start with, somewhere something happened and as a result the train in direction 2 was able to depart the station eighth time units past the scheduled departure time. When the trains keep travelling with their usual speed, the trains in direction 1 and 3 will also depart the station too late in the next step $k = 2$ (shown by $z(2)$ in the first and third element). This is because these two trains have to wait for the arrival of the originally delayed train before they are allowed to depart. To limit the effect of the originally delayed train on these two trains, we let the train in direction 2 speed up to catch up some of the delay. As a consequence, the delay of the departure times of the trains in direction 1 and 3 at $k = 2$ are reduced by 2 (shown in $x(2)$ in the first and third element). At $k = 2$, train 2 and 4 can depart as usual, because the trains to which they are connected (train 2 is connected to train 3 and 4 and train 4 is connected to train 3 and itself) were not delayed at $k = 1$. These two trains also did not notice that the train in direction 2 drove faster, because they are not connected to that train. In the next steps approximately the same happens but for different trains.

To conclude, when one train has a delay, all trains connected to that train also have delayed departure times, except when the delay is absorbed by slack time. Speeding up a train that departed too late can help reduce the delay of the departure times of the connected trains. The trains that are not connected to a delayed train can depart as usual. The departure times of the connected trains will not alter, when the speeding leads to an earlier arrival than scheduled. The trains that are not connected to the train that speeds up, do not experience any influence of the speeding.

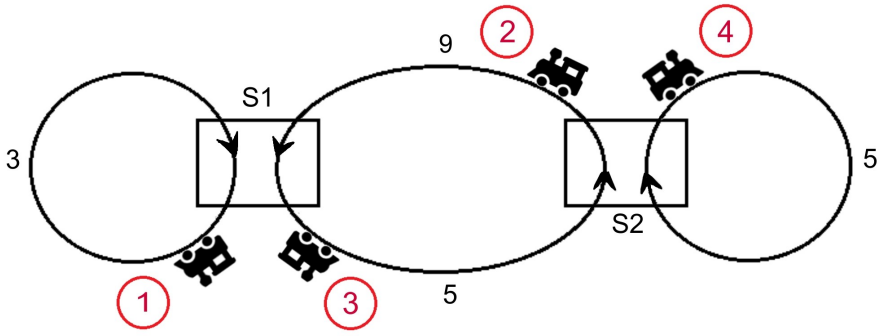

Figure 3: A simple fictional railway network with the different directions labeled in red.

4.3 Control strategy: Breaking connections

The second control strategy, that will be discussed in this thesis, is to break certain connections between trains. In some situations, it can be a more efficient choice to break a connection. In general, these are the connections that allow a passenger changeover at a station. For example, if the train in direction 2 has a large delay, it is inconvenient that the train in direction 1 should keep waiting for the arrival of that train in order to enable a passenger changeover between both trains at station 1. If we break the connection between these trains, the train in direction 1 can keep running according to timetable. Other connections cannot be broken. For example, the train in direction 3 should still wait for the arrival of the train in direction 2, because a train cannot depart before it has arrived.

To find the control strategy that indicates which connections should be broken, the theory of [De Vries et al., 1998] is used. In the next subsection this control strategy will be implemented in the switching max-plus model for speeding up trains. First, a decision variable u_{il} is introduced, that indicates whether the kth train in direction i will wait for the connecting train in direction l.

$$u_{il} = \begin{cases} 0 \text{ if } i \text{ will wait for } l \\ \varepsilon \text{ otherwise} \end{cases} . \tag{4.8}$$

The goal is to choose the u_{il} such that the delays of all trains are minimal and at the same time as many connections as possible will be maintained. The chosen u_{il} will define the control strategy used in the SMPL model.

The u_{il}'s are chosen such that the objective function J is minimal, with

$$J = \frac{(\sum_k \sum_j z_j(k))^\alpha}{1 + \sum_k \sum_{i,l} u'_{il}(k)}. \tag{4.9}$$

The numerator denotes the total delay, with $z(k) = x(k) - d(k)$. The total delay must be as small as possible. The denominator denotes the number of connections, where

$$u'_{il}(k) = \begin{cases} 1 \text{ if } u_{il} = 0 \\ 0 \text{ if } u_{il} = \varepsilon \end{cases}$$

and the 1 in the denominator is added to prevent that the denominator becomes 0. We want the number of connections to be as high as possible. The α indicates which of the objectives is more important.

To find the best control strategy (that leads to a minimal J), first the set U of possible control variables is determined. Only the unnecessary connections are considered. In our railway network, these are u_{12}, u_{31}, u_{43} and u_{24}. The possible $u_{ij}(k)$'s are the controls that correspond to delayed trains. These are the controls that give $a_{ij} \otimes x_j(k) \otimes u_{ij}(k) > d_i(k+1)$ when $u_{ij} = 0$, starting from the k of the initial delay. Only these controls can influence the total delay in the system. To find these $u_{ij}(k)$ a Python program is written that can be

found in appendix A. Once the controls are determined, different control strategies can be composed by setting one or more elements of U equal to ε, meaning that the connection is broken. For each of these control strategies the J can be calculated. The strategy for which J has the smallest value will be chosen to include in the switching model. An example of how to find the optimal control strategy will be given in subsection 4.3.2.

4.3.1 Switching max-plus-linear model for both strategies combined

The controls will be added to the SMPL model of the previous control strategy, in which we let one train run faster to catch up the delay. The first mode is again the original MPL system, the second mode contains the altered system matrix in which we let one train run faster. Additionally, mode 2 contains the broken connections determined by the method in the article of [De Vries et al., 1998].

Mode 1

$$x(k) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k). \tag{4.10}$$

Mode 2

$$\begin{aligned} &x(k) = A \otimes x(k-1) \oplus d(k), \\ &\text{where } [A]_{ij} = [S]_{ij} \otimes u_{ij}(k-1). \end{aligned} \tag{4.11}$$

In mode 2, the matrix S is the matrix where the train in direction 2 runs faster,

$$S = \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix}$$

and $u_{ij}(k) = \begin{cases} 0 \text{ if } i \text{ will wait for } j \\ \varepsilon \text{ otherwise} \end{cases}$.

The switching variable $z(k)$ is again defined by

$$z(k) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1) \oplus d(k) \in \mathbb{R}^4_{\max} \tag{4.12}$$

and

$$\mathcal{Z}^1 = \{z(k) |\ z(k) = d(k)\}, \tag{4.13}$$

$$\mathcal{Z}^2 = \{z(k) |\ \exists\ i \in \{1,2,3,4\} :\ z_i(k) > d_i(k)\}. \tag{4.14}$$

4.3.2 Example of a delay

The same delay as in example 1 of the previous section is analysed. First we need to determine the set U. Hence we need to know which of the unnecessary connections, $u_{12}(k),\ u_{31}(k),\ u_{43}(k),\ u_{24}(k)$, give $a_{ij} \otimes x_j(k) \otimes u_{ij}(k) > d_i(k+1)$ when $u_{ij} = 0$, starting from $k = 1$. This is programmed in Python and can be found in appendix A. The controls found for this initial delay are

$$u_{12}(1),\ u_{12}(3),\ u_{43}(2) \text{ and } u_{24}(3).$$

Now the different control strategies can be composed by setting these equal to 0 or ε. The number of possible control strategies is 2^n, with n the number of controls found. In this case there are 2^4 control strategies possible, these are given in Table 1.

	$u_{12}(1)$	$u_{12}(3)$	$u_{43}(2)$	$u_{24}(3)$
1	0	0	0	0
2	ε	0	0	0
3	0	ε	0	0
4	ε	ε	0	0
5	0	0	ε	0
6	ε	0	ε	0
7	0	ε	ε	0
8	ε	ε	ε	0
9	0	0	0	ε
10	ε	0	0	ε
11	0	ε	0	ε
12	ε	ε	0	ε
13	0	0	ε	ε
14	ε	0	ε	ε
15	0	ε	ε	ε
16	ε	ε	ε	ε

Table 1: Possible control strategies.

For each of these control strategies, we calculate the value of the objective function (4.9) for $\alpha = 1$ and $\alpha = \frac{1}{2}$. When $\alpha = \frac{1}{2}$, maximizing the number of connections is more important than minimizing the total delay. The total delay is calculated as in section 4.1, but when $u_{ij}(k) = \varepsilon$, the $[A]_{ij}$ is set to ε in the MPL model. This leads to the following results

	$\sum u'$	$\sum z$	J_1	$J_{\frac{1}{2}}$
1	4	22	4,4	0,938
2	3	17	4,25	1,031
3	3	21	5,25	1,146
4	2	16	5,333	1,333
5	3	16	4	1
6	2	11	3,667	1,106
7	2	15	5	1,291
8	1	10	5	1,581
9	3	21	5,25	1,146
10	2	16	5,333	1,333
11	2	20	6,667	1,491
12	1	15	7,5	1,936
13	2	16	5,333	1,333
14	1	11	5,5	1,658
15	1	15	7,5	1,936
16	0	10	10	3,162

Table 2: Value of J for different control strategies.

We conclude that when $\alpha = 1$, control strategy 6 is best. That is when $u_{12}(1) = \varepsilon$ and $u_{43}(2) = \varepsilon$. When $\alpha = \frac{1}{2}$, control strategy 1 is best. Hence when the focus lays on maximizing the number of connections, it turns out that breaking no connections is best. This leads to the example in section 4.2 when the SMPL model is applied. When control strategy 6 is applied, the total delay in the system reduces to 11 and the SMPL model takes the following steps

$$x(1) = \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix}$$

$$z(2) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 22 \\ 20 \\ 22 \\ 20 \end{pmatrix} = \begin{pmatrix} 29 \\ 20 \\ 29 \\ 20 \end{pmatrix} \in \mathcal{Z}^2$$

$$x(2) = \begin{pmatrix} 5 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 12 \\ 18 \\ 12 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 22 \\ 20 \\ 22 \\ 20 \end{pmatrix} = \begin{pmatrix} 22 \\ 20 \\ 27 \\ 20 \end{pmatrix}$$

$$z(3) = \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 22 \\ 20 \\ 27 \\ 20 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 32 \\ 34 \\ 32 \\ 34 \end{pmatrix} \in \mathcal{Z}^2$$

$$
\begin{aligned}
x(3) &= \begin{pmatrix} 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 9 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 7 \end{pmatrix} \otimes \begin{pmatrix} 22 \\ 20 \\ 27 \\ 20 \end{pmatrix} \oplus \begin{pmatrix} 32 \\ 30 \\ 32 \\ 30 \end{pmatrix} = \begin{pmatrix} 32 \\ 34 \\ 32 \\ 30 \end{pmatrix} \\
&\vdots \\
z(5) &= \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 43 \\ 40 \\ 43 \\ 40 \end{pmatrix} \oplus \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} = \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} \in \mathcal{Z}^1 \\
x(5) &= \begin{pmatrix} 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \\ 5 & 11 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 43 \\ 40 \\ 43 \\ 40 \end{pmatrix} \oplus \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix} = \begin{pmatrix} 52 \\ 50 \\ 52 \\ 50 \end{pmatrix}.
\end{aligned}
$$

At $k = 2$, the system switches to mode 2. This mode gives $[A]_{12} = [S]_{ij} \otimes u_{12}(1) = 9 \otimes \varepsilon = \varepsilon$. At $k = 3$, another connection is broken, namely $[A]_{43} = [S]_{43} \otimes u_{43}(2) = 7 \otimes \varepsilon = \varepsilon$. Breaking these connections leads to a smaller total delay. However, the number of steps it takes to solve the delay stays the same.

5 Modelling of a larger railway network

The theory discussed in the previous chapters will be applied to a larger, more realistic railway network. The railway network is a modification of the railway network discussed in chapter 8 of the book of [Heidergott et al., 2006]. Our version consists of three long-distance lines that connect the Dutch cities Amsterdam (Asd), Amersfoort (Amf), Deventer (Dv) and Zwolle (Zl). On these lines intercity trains run back and forth. The railway network is shown in Figure 4. The three different intercity lines are colored, the ten different directions are labeled in red, the connections are displayed with blue arrows and the travel times in minutes are shown for each track. The travel times also include waiting times and transfer times. The tracks in direction 3 and 7 are large circuits that pass other stations as well. The tracks between these stations are combined to one track with a large travel time. Track 3 goes all the way to the Dutch city Groningen and track 7 goes to 's Hertogenbosch and back.

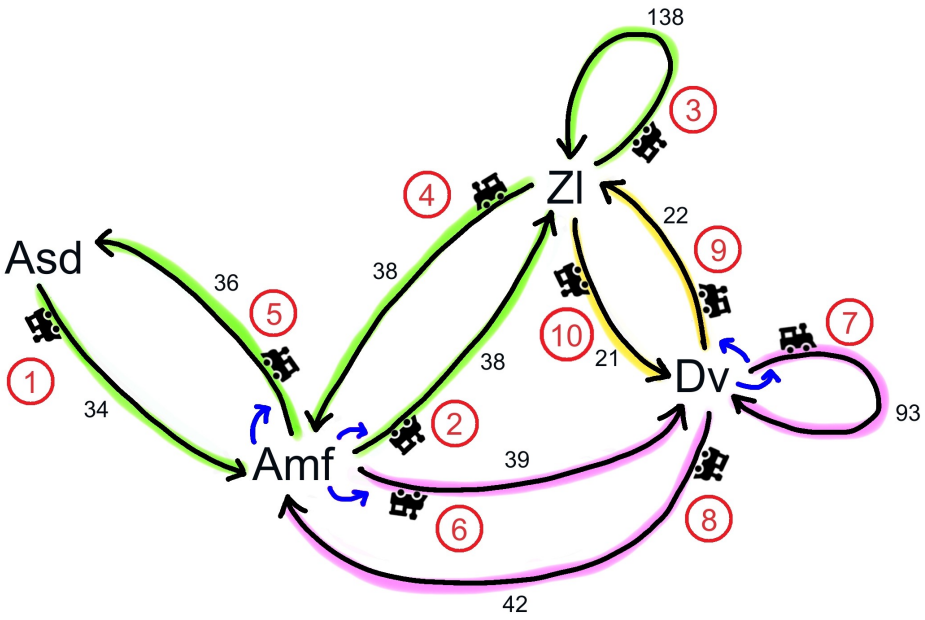

Figure 4: A larger, more realistic railway network with the different directions labeled in red. The connections between the three different intercity lines are displayed with blue arrows.

This railway network differs from the example in the book of [Heidergott et al., 2006]. Some line segments are left out and some are combined to one track. That is because the modelling of the example in the book of [Heidergott et al., 2006] requires additional techniques that are not in the scope of this thesis. Within our smaller version of the railway network in the book, we assume the same travel times and connections as the example in the book. Remark that the assumptions made in this chapter do not exactly correspond to reality, but they are plausible. Other situations can be modelled similarly.

The first step is to design the max-plus-linear system and to design a train timetable

that serves as input of the system. The following equations for the ten directions can be made:

$$\begin{aligned}
x_1(k) &= a_{15} \otimes x_5(k-1),\\
x_2(k) &= a_{21} \otimes x_1(k-1) \oplus a_{28} \otimes x_8(k-1),\\
x_3(k) &= a_{32} \otimes x_2(k-1),\\
x_4(k) &= a_{43} \otimes x_3(k-1),\\
x_5(k) &= a_{54} \otimes x_4(k-1) \oplus a_{58} \otimes x_8(k-1),\\
x_6(k) &= a_{61} \otimes x_1(k-1) \oplus a_{68} \otimes x_8(k-1),\\
x_7(k) &= a_{76} \otimes x_6(k-1) \oplus a_{7,10} \otimes x_{10}(k-1),\\
x_8(k) &= a_{87} \otimes x_7(k-1),\\
x_9(k) &= a_{97} \otimes x_7(k-1) \oplus a_{9,10} \otimes x_{10}(k-1),\\
x_{10}(k) &= a_{10,9} \otimes x_9(k-1).
\end{aligned} \tag{5.1}$$

The equations in (5.1) lead to a MPL system of the form

$$x(k) = \begin{pmatrix}
\varepsilon & \varepsilon & \varepsilon & \varepsilon & 36 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
34 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\
\varepsilon & 38 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & 138 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & 38 & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\
34 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 39 & \varepsilon & \varepsilon & \varepsilon & 21 \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 93 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 93 & \varepsilon & \varepsilon & 21 \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 22 & \varepsilon
\end{pmatrix} \otimes x(k-1) \oplus d(k). \tag{5.2}$$

The train time table for this railway network is again composed by means of the power algorithm. The Python code used in the previous chapters can be used for this. The eigenvalue of the state matrix is 58. Thus the constant interdeparture time found for this railway network is 58, meaning that every 58 minutes the intercity trains can leave in each direction. For the timetable we will turn this into 60 minutes, meaning every hour. The eigenvector corresponding to the eigenvalue of 58 is $(38, 20, 0, 80, 60, 20, 1, 36, 36, 0)^{\mathrm{T}}$. Suppose we let the timetable start at 5:00 AM, then the train timetable will be as in Table 3. In equation (5.2) is calculated with minutes, so that for example the $d_1(k)$ used in the equation are consecutively 38, 38+60=98, 98+60=158, etc. To display the results, the values will be converted to clock times.

Train	d(0)	d(1)	d(2)	etc.
1	5:38	6:38	7:38	...
2	5:20	6:20	7:20	...
3	5:00	6:00	7:00	...
4	6:20	7:20	8:20	...
5	6:00	7:00	8:00	...
6	5:20	6:20	7:20	...
7	5:01	6:01	7:01	...
8	5:36	6:36	7:36	...
9	5:36	6:36	7:36	...
10	5:00	6:00	7:00	...

Table 3: The train timetable for the railway network.

Now the propagation of delays through this network can be investigated. Suppose the train in direction 8 has an initial delay of 12 minutes. Hence $x(0) = (38, 20, 0, 80, 60, 20, 1, 48, 36, 0)^{\mathrm{T}}$. By applying the MPL model (5.2), we find that at $k = 6$ the delay is out of the system and the total delay is 76 minutes.

Next, the effect of the control strategies is investigated by applying the switching max-plus-linear models. In our railway network we assume that the train in direction 3 is allowed to reduce its travel time to 134, the train in direction 7 may reduce its travel time to 89 and the trains in direction 2 and 4 may reduce their travel time to 36. If we call the state matrix in (5.2) matrix A^1, the SMPL model for the strategy of faster running trains becomes:

Control strategy: Let trains run faster

Mode 1

$$x(k) = A^1 \otimes x(k-1) \oplus d(k), \tag{5.3}$$

Mode 2

$$x(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 36 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 34 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\ \varepsilon & 36 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 134 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 36 & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\ 34 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 42 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 39 & \varepsilon & \varepsilon & \varepsilon & 21 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 89 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 89 & \varepsilon & \varepsilon & 21 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 22 & \varepsilon \end{pmatrix} \otimes x(k-1) \oplus d(k), \tag{5.4}$$

Switching variable

$$z(k) = A^1 \otimes x(k-1) \oplus d(k) \in \mathbb{R}^{10}_{\max}, \tag{5.5}$$

$$\mathcal{Z}^1 = \{z(k)|\ z(k) = d(k)\}, \tag{5.6}$$
$$\mathcal{Z}^2 = \{z(k)|\ \exists\ i \in \{1, 2, \ldots, 10\} :\ z_i(k) > d_i(k)\}. \tag{5.7}$$

When applying this SMPL model to the delay, the total delay in the system reduces from 76 minutes to 38 minutes (excluding the initial delay) and the delay is out of the network at $k = 4$, compared to $k = 6$ when there is no intervention. The train timetable will change to Table 4.

Train	d(0)	d(1)	d(2)	d(3)	d(4)
1	5:38	6:38	7:38	8:38	9:38
2	5:20	6:20+10	7:20	8:20	9:20
3	5:00	6:00	7:00+6	8:00	9:00
4	6:20	7:20	8:20	9:20	10:20
5	6:00	7:00	8:00	9:00	10:00
6	5:20	6:20+10	7:20	8:20	9:20
7	5:01	6:01	7:01+8	8:01	9:01
8	5:36+12	6:36	7:36	8:36+2	9:36
9	5:36	6:36	7:36	8:36+2	9:36
10	5:00	6:00	7:00	8:00	9:00

Table 4: The train timetable for the railway network when train 8 is delayed. As a consequence certain trains will drive faster.

The switching max-plus-linear model in which speeding up trains and breaking connections is combined is the following:

Control strategy: Let trains run faster and break certain connections

Mode 1

$$x(k) = A^1 \otimes x(k-1) \oplus d(k), \tag{5.8}$$

Mode 2

$$\begin{aligned} &x(k) = A^2 \otimes x(k-1) \oplus d(k), \\ &\text{where } [A^2]_{ij} = [S]_{ij} \otimes u_{ij}(k-1). \end{aligned} \tag{5.9}$$

In mode 1, A^1 is the state matrix as in equation (5.2). In mode 2, the matrix S is the matrix where four trains run faster, which is the state matrix in equation (5.4). Also, $u_{ij}(k) = \begin{cases} 0 \text{ if } i \text{ will wait for } j \\ \varepsilon \text{ otherwise} \end{cases}$.

Switching variable

$$z(k) = A^1 \otimes x(k-1) \oplus d(k) \in \mathbb{R}^{10}_{\max}, \tag{5.10}$$

$$\mathcal{Z}^1 = \{z(k)|\ z(k) = d(k)\}, \tag{5.11}$$
$$\mathcal{Z}^2 = \{z(k)|\ \exists\ i \in \{1, 2, \ldots, 10\} :\ z_i(k) > d_i(k)\}. \tag{5.12}$$

To determine the controls u_{ij} that can influence the delay, the same Python code as in the previous chapters is used. The unnecessary connections in this network are $u_{58}(k)$, $u_{61}(k)$, $u_{28}(k)$, $u_{97}(k)$, and $u_{7,10}(k)$. The controls found by the Python code, that can influence this delay, are $u_{28}(0)$ and $u_{97}(2)$. This leads to $2^2 = 4$ different control strategies for breaking connections. For each of the control strategies the value of the objective function J (4.9) is calculated. The results are shown in Table 5.

	$u_{28}(0)$	$u_{97}(2)$	$\sum u'$	$\sum z$	J_1	$J_{\frac{1}{2}}$
1	0	0	2	38	12.667	2.055
2	ε	0	1	22	11	2.345
3	0	ε	1	36	18	3
4	ε	ε	0	20	20	4.472

Table 5: The control strategies for breaking connections and the values for the objective function J, with $\alpha = 1$ and $\alpha = \frac{1}{2}$.

When $\alpha = 1$ (minimizing the total delay and maximizing the number of connections are equally important), control strategy 2 is best. When $\alpha = \frac{1}{2}$ (maximizing the number of connections is more important), control strategy 1 is best. Control strategy 1 leads to the same results as in Table 4. When control strategy 2 is applied, the total delay in the railway network reduces to 22 minutes. At $k = 4$, the delay is out of the network. The train timetable changes to Table 6.

Train	d(0)	d(1)	d(2)	d(3)	d(4)
1	5:38	6:38	7:38	8:38	9:38
2	5:20	6:20	7:20	8:20	9:20
3	5:00	6:00	7:00	8:00	9:00
4	6:20	7:20	8:20	9:20	10:20
5	6:00	7:00	8:00	9:00	10:00
6	5:20	6:20+10	7:20	8:20	9:20
7	5:01	6:01	7:01+8	8:01	9:01
8	5:36+12	6:36	7:36	8:36+2	9:36
9	5:36	6:36	7:36	8:36+2	9:36
10	5:00	6:00	7:00	8:00	9:00

Table 6: The train timetable for the railway network when train 8 is delayed. As a consequence certain trains will drive faster. Additionally, the connection between train 8 and train 2 will be broken at the first step.

To conclude, the total delay of 76 minutes can be reduced to 22 minutes when the trains run faster and one connection is broken. The models provided in this chapter can calculate exactly how the delay propagates through the railway network and provide passengers information about the new departure times.

6 Conclusion

To summarize, the goal of the thesis was to model train delays and to model possible control strategies to solve these delays. First a basic understanding of max-plus algebra is obtained, where also max-plus-linear (MPL) systems and switching max-plus-linear (SMPL) systems are explained. Then a simple railway network is constructed, for which a desirable train timetable is designed using the power algorithm. The departure times of each event step in this railway network are modelled into a MPL system. Subsequently, delays are executed on the departure times and the propagation of these delays is modelled. The delays are solved by applying two control strategies: let certain trains run faster and break unnecessary connections. Both strategies are modelled into a switching max-plus-linear model to automate the system. Finally, all required theory is applied at a larger, more realistic railway network.

In the examples it can be observed that the total delay in the system reduced drastically by letting certain trains run faster. The delay was reduced some more by additionally breaking connections. It can be concluded that the switching max-plus-linear models obtained in this thesis, provide train operators a more automated way to decide how to intervene when a delay is detected. This has often been done more intuitively. The models in this thesis provide methods to calculate exactly how the delay propagates through the network when certain control strategies are applied and, based on that, decisions can be made quicker. Moreover, it is possible to calculate the consecutive departure times so that the passengers can be informed quickly about the new departure times as a consequence of the delay and how long it will take for the trains to run according to timetable again.

This thesis adds the modelling of faster running trains to existing literature. In general, slowing down the trains does not require additional modelling methods, since it is always possible to let a train wait a little longer before a railway signal for instance. However, to let a train run faster is less straight forward. As we have seen in this thesis, speeding up trains is also a control strategy to solve delays and can be modelled systematically.

7 Discussion

A few remarks on this thesis can be made that may lead to further research of this topic.

First of all, in our SMPL models the altered matrix, where some trains run faster, is used in every step until the delay is out of the system. However, it may be safer to speed up as few times as possible. So only when the train that is allowed to speed up has a delayed departure time, we let that train speed up. In the steps in which trains that are not allowed to speed up have delayed departure times, the regular state matrix can be used, since the altered matrix has no effect on the delay. This can be noticed in the example of subsection 4.2.2. Here $z(3)$ and $x(3)$ have the same outcome, hence it was actually not necessary to speed up the train in direction 2. When this train unnecessarily speeds up, it has a higher risk of causing accidents, while at the end station it just stands still because it arrived too early. Also, when multiple trains in the network are allowed to speed up, not all of them need to speed up in every step to solve the delay. This can be adjusted in the SMPL model.

Secondly, when a train has an extremely large delay, it may be more convenient to replace that train by a bus. Otherwise the necessary connected trains keep waiting on that train, because these connections cannot be broken. Then a new MPL model can be constructed such that the departure times are synchronized with the bus. This can be added as a mode to the SMPL model.

Moreover, the models do not automate the decision making completely. To determine which control strategy for breaking connections one wants to implement into the SMPL model, humans need to choose what is more important: minimizing the total delay or maximizing the number of maintained connections. This choice may depend on the situation.

Further remarks mainly relate to the appliance of these methods to an even larger, realistic railway network. To start with, including all tracks in the model leads to an extremely large state matrix. It also takes a while to set up all separate equations to form the matrix. Solutions for this should be investigated. Moreover, the number of possible control strategies to be analysed for breaking connections can blow up. A solution for this is suggested in the article of [De Vries et al., 1998]. Furthermore, the Python program can be written more efficiently or another programming language can be used, such that the program's running time will be reduced for a large network.

The last remark is that these models do not limit to trains. We can, for instance, connect a tram or a bus to the train network and include a suitable transfer time for passengers to changeover from train to bus or the other way around. This can be modelled the exact same way.

References

[De Vries et al., 1998] De Vries, R., De Schutter, B., and De Moor, B. (1998). On max-algebraic models for transportation networks. pages 457–462, Cagliari, Italy. Proceedings of the 4th International Workshop on Discrete Event Systems (WODES'98).

[Heidergott et al., 2006] Heidergott, B., Olsder, G. J., and Van der Woude, J. (2006). *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press.

[Kalamboukis, 2018] Kalamboukis, V. P. (2018). Matrix spans in max-plus algebra and a graph-theoretic approach to switching max-plus linear systems. Master's thesis, TU Delft.

[Kersbergen, 2015] Kersbergen, B. (2015). *Modeling and Control of Switching Max-Plus-Linear Systems: Rescheduling of railway traffic and changing gaits in legged locomotion*. PhD thesis, TU Delft.

[Van den Boom and De Schutter, 2012] Van den Boom, T. J. J. and De Schutter, B. (2012). Modeling and control of switching max-plus-linear systems with random and deterministic switching. *Discrete Event Dynamic Systems: Theory and Applications*, 22(3):293–332.

[Van den Muijsenberg, 2015] Van den Muijsenberg, M. D. (2015). Scheduling using max-plus algebra: General framework and application to a baggage handling system. Master's thesis, TU Delft.

Appendix

A Python code for calculations during the thesis

```
import numpy as np

# Max-plus algebra: x = +, + = max

# function A matrix
eps=-float("inf")
def Amat(T1,T2,R11,R22,R12,R21):
    A=np.full((4,4),eps)
    A[0][0]=T1+R11
    A[0][1]=T2+R12
    A[1][2]=T1+R21
    A[1][3]=T2+R22
    A[2][0]=T1+R11
    A[2][1]=T2+R12
    A[3][2]=T1+R21
    A[3][3]=T2+R22
    return A


# function matrix multiplication
def mult(A,B):
    res=np.zeros((len(A),len(B[0])))
    lst=[]
    n=0
    m=len(B)
    for i in range(len(A)):
        for k in range(len(B[0])):
            for j in range(len(B)):
                lst.append(A[i][j]+B[j][k])
    for i in range(len(A)):
        for j in range(len(B[0])):
            res[i][j]=max(lst[n:m])
            n=n+len(B)
            m=m+len(B)
    return res

# function matrix power
def matpow(A,power):
    if power==1:
        B=A
    if power>1:
        B=mult(A,matpow(A,power-1))
    return B

# Power algorithm: departures from S1 and S2
def powalg(A,x):
    lst=[x]
```

```
48      v=[]
49      eigv=np.zeros((len(x),1))
50      for i in range(len(lst)):
51          for j in range(len(x)-1):
52              while lst[-1][j][0]-lst[i][j][0]!=lst[-1][j+1][0]-lst[i][j
                    +1][0] or lst[-1][0][0]-lst[i][0][0]==0:
53                  lst.append(mult(A,lst[-1]))
54              else:
55                  c=lst[-1][0][0]-lst[i][0][0]
56                  p=len(lst)-1
57                  q=i
58      lamb=c/(p-q)
59      for j in range(1,p-q+1):
60          v.append(lamb*(p-q-j)+lst[q+j-1])
61      for k in v:
62          for i in range(len(k)):
63              if k[i][0]>eigv[i][0]:
64                  eigv[i][0]=k[i][0]
65
66      return lamb,eigv-min(eigv)[0]
67
68
69
70  ##############################################################################
71  A=Amat(2,2,3,5,9,5) # A matrix
72  x=np.zeros((4,1))
73  dly=np.array([[12],[18],[12],[10]])
74  k=1
75
76  print 'A='+str(A)
77  print 'Eigenvalue = '+str(powalg(A,x)[0])
78  print 'Eigenvector = '+str(powalg(A,x)[1])
79
80  # Desirable timetable d(k)
81  d=[powalg(A,x)[1]]
82  for i in range(20):
83      d.append(d[i]+powalg(A,x)[0]+1)
84
85  print('Timetable = '+str(d))
86
87  # Delayed departure times without intervening (solved by slack time)
88  def delay(dly,k,A):
89      x=[dly]
90      while (x[-1]>d[k]).any():
91          x.append(mult(A,x[-1]))
92          k=k+1
93          # Do not leave before departure time in timetable:
94          for i in range(len(dly)):
95              if x[-1][i]<=d[k][i]:
96                  x[-1][i]=d[k][i]
97      else:
98          return x
99
100 # Propagation of delay z(k)
101 def pod(k,A):
```

```
102     z=[]
103     n=k
104     for i in range(len(delay(dly,k,A))):
105         z.append(delay(dly,k,A)[i]-d[n])
106         n=n+1
107     return z
108 
109 print 'Delayed departure time is '+str(dly)+' at step '+str(k)
110 print 'Delayed departure times from moment of first delay = '+str(delay(dly
    ,k,A))
111 print 'Delay is '+str(pod(k,A))
112 print 'Total delay: '+str(sum(sum(pod(k,A))-pod(k,A)[0])[0])
113 print 'After '+str(len(delay(dly,k,A))-1)+' steps delay is solved by buffer
    '
114 
115 # Departure times with faster A matrix
116 A_inhaal=Amat(2,2,3,5,7,5)
117 print 'Catch up timetable '+str(delay(dly,k,A_inhaal))
118 print 'After '+str(len(delay(dly,k,A_inhaal))-1)+' steps delay is solved'
119 print 'Total delay: '+str(sum(sum(pod(k,A_inhaal))-pod(k,A_inhaal)[0])[0])
120 
121 ##############################################################################
122 # Breaking connections
123 # Find controls
124 def ctrl():
125     u_12=[]
126     u_31=[]
127     u_43=[]
128     u_24=[]
129     for i in range(len(delay(dly,k,A_inhaal))-2):
130         if A_inhaal[0][1]+delay(dly,k,A_inhaal)[i][1]>d[k+1+i][0]:
131             u_12.append(k+i)
132         if A_inhaal[2][0]+delay(dly,k,A_inhaal)[i][0]>d[k+1+i][2]:
133             u_31.append(k+i)
134         if A_inhaal[3][2]+delay(dly,k,A_inhaal)[i][2]>d[k+1+i][3]:
135             u_43.append(k+i)
136         if A_inhaal[1][3]+delay(dly,k,A_inhaal)[i][3]>d[k+1+i][1]:
137             u_24.append(k+i)
138     return u_12,u_31,u_43,u_24
139 
140 print 'u_12('+str(ctrl()[0])+'), u_31('+str(ctrl()[1])+'), u_43('+str(ctrl
    ()[2])+'), u_24('+str(ctrl()[3])+')'
141 
142 
143 # control strategies
144 def combs(places):
145     if len(places)==0:
146         return [[]]
147     c=[]
148     for i in combs(places[1:]):
149         c+=[i, i+[places[0]]]
150     return c
151 
152 def strat():
153     l=[]
```

```
154     for i in ctrl():
155         l.append(len(i))
156     S=np.zeros((2**sum(l), sum(l)+1))
157     for i in range(len(S)):
158         S[i][0]=i+1
159         for j in combs(range(sum(l)))[i]:
160             S[i][j+1]=eps
161     return S
162 
163 print strat()
164 
165 z=[22,17,21,16,16,11,15,10,21,16,20,15,16,11,15,10]
166 def table(z):
167     tab=np.zeros((16,5))
168     for i in range(16):
169         tab[i][0]=i+1
170         for j in range(1,5):
171             if strat()[i][j]==0:
172                 tab[i][1]=tab[i][1]+1
173         tab[i][2]=z[i]
174         tab[i][3]=float(z[i])/float((tab[i][1]+1))
175         tab[i][4]=float(z[i])**(0.5)/float((tab[i][1]+1))
176     return tab
177 
178 print table(z)
```