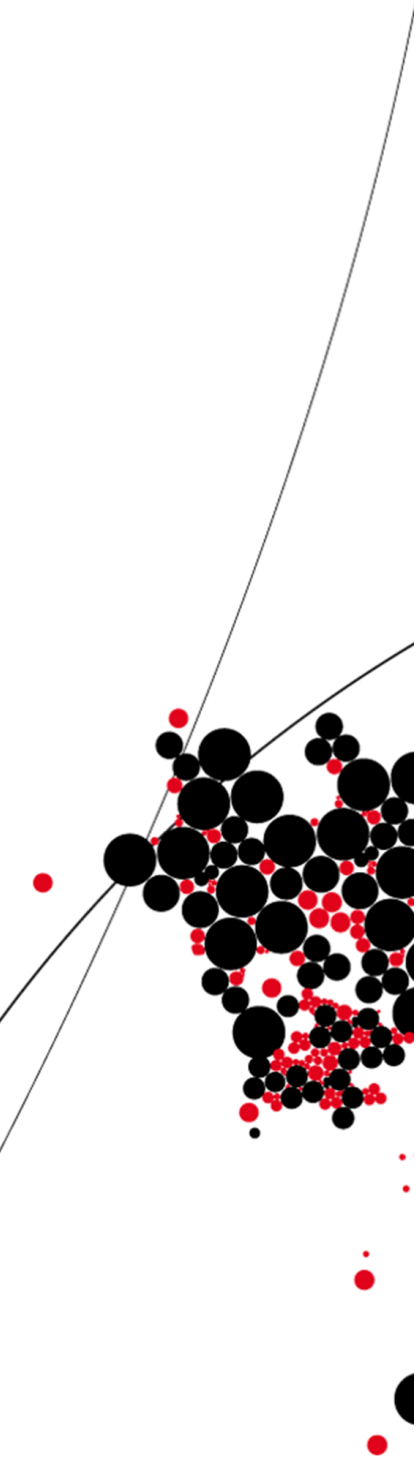
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science



State Space Identification and Minimal State Space Realization of Max-Plus Linear Systems

Sven Stienissen

M.Sc. Thesis Applied Mathematics - Mathematics of Data Science
June 2023

Supervisors:
Dr. H. Hang (UT)
Prof. dr. ir. A.A. Basten (TU/e)
Dr. Y. Li (TNO-ESI)

Graduation committee:
Dr. H. Hang (UT)
Prof. dr. A.J. Schmidt-Hieber (UT)
Dr. A. Skopalik (UT)

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Acknowledgement

This thesis completes my master's program in Applied Mathematics at the University of Twente which I started in September 2020 after retrieving my Bachelor's diploma in Industrial Engineering & Management. For the past 8 months, starting from November 2022, I have been working really hard on this project, encountering various challenges. The successful completion of this thesis would not have been possible without the generous assistance of many people.

First of all, I would like to thank Yonghui Li for all the invaluable conversations that we had. You were always available to discuss any problems I encountered during the process and were always keen to give me some new insights. In addition, you introduced some practical problems to me very clearly, which helped me a lot to put this research into perspective. Furthermore, I would like to express my gratitude to Twan Basten for all the interesting discussions that we had. You gave me a lot of valuable feedback during our meetings and were always able to take a helicopter view on this research to find the value in this thesis.

Additionally, I would like to extend my gratitude to Hanyuan Hang, my supervisor from the University of Twente, for his feedback, and flexibility to meet. Also, I would like to thank my graduation committee consisting of Hanyuan Hang, Johannes Schmidt-Hieber, and Alexander Skopalik.

Finally, I want to acknowledge the support, motivation, and occasional distraction from my friends and family throughout this period. I want to express my special thanks to Naèla for all her support. I look back with great pleasure on my time as a graduate intern at TNO-ESI and was able to learn a lot during the past 8 months.

Sven Stienissen

Abstract

We present a method to identify the parameters of a state space model for a max-plus linear system based on the data from input-output sequences. This method is based on modeling the system as a mixed-integer program. We show that this method is computationally more efficient compared to existing methods, given the assumption that the system and data are not corrupted by noise. Furthermore, we present a linear program that could be used for state space identification in certain cases. This method is even more computationally efficient and allows us to identify max-plus linear systems of a higher order. Additionally, we will show that the same mixed-integer programming formulation can be used to find the minimal state space realization of a max-plus linear system and present an algorithm to find this minimal realization. Moreover, we will present a method to model capacity constraints in a max-plus linear system.

Contents

Chapter 1

Introduction

1.1 Background

In the modern high-tech industry, with the ever increasing and changing market needs, there is a need for smart and flexible manufacturing systems that can productively handle various types of products, such that factories can quickly satisfy these market needs and maximize profits. Manufacturing systems often consist of multiple coupled sub-systems, each consisting of one or more machines, that perform operations on the products being produced. Often these sub-systems are created by different companies and are thus not always perfectly optimized to cooperate, which could result in a loss of productivity. Additionally, sometimes the exact behaviour of a sub-system is completely unknown. To meet customer needs, companies must have reliable estimates of their order delivery times, which in turn requires accurate predictions of the finishing times of the jobs on the manufacturing system. However, given the complex nature of operations within these systems, and the unknown behaviours, estimating finishing times can be a complex problem. In order to do this, it will be key to discover the behaviour of the unknown system dynamics. This can be seen as learning a black-box or grey-box model of the unknown sub-system.

In literature, multiple methods have been proposed to model and analyse manufacturing systems, such as queuing models, Petri nets, and discrete event systems (DES). In general, DES lead to a nonlinear description in conventional algebra. However, there exists a subclass of DES for which this model becomes “linear” when it is formulated in the max-plus algebra [1], [5], where the conventional operators addition and multiplication are replaced by maximization and addition, respectively. This subclass of DES has two important characteristics, namely synchronization and no concurrency or choice. Synchronization requires the availability of several resources or users at the same time, whereas concurrency appears, for instance, when some users must choose among several resources, at a certain time [1]. The maximization

operator corresponds to synchronization, where a new operation starts as soon as all preceding operations have been finished. The addition operator corresponds to the duration of an activity, where the finishing time is equal to the starting time plus the duration. DES with synchronization and no concurrency are known as *max-plus linear (MPL) systems* [11], [33]. In addition to modeling production systems, the max-plus algebra can be used to model railroad networks, array processors, queuing systems, and urban traffic networks [1], [5], [19].

In order to use MPL systems to estimate the behaviour of a system, we need to be able to estimate the parameters and structure of the model. In literature, this problem is known as system identification [24]. Multiple methods have been proposed to estimate the parameters of MPL systems [13], [27], [32]. However, most of these methods assume that the internal structure of the system is completely known, and that the state is assumed to be measurable. In this thesis, we assume that we can only observe input-output sequences of the system. Input-output sequences are a collection of the times that some item enters and leaves the system.

Furthermore, many modern mathematical models that are being used to represent real-world processes present difficulties in numerical calculations due to their complexity, and size or dimension of the model. Particularly, issues arise when these models need to be used in real-world applications where there is only a limited time for computations, as in smart and flexible manufacturing systems. One of the methods proposed in literature is to find a minimal realization of the system. The minimal realization problem deals with finding the smallest possible model that accurately represents the observable data of a given system. The data are typically the impulse response of the system, the step response, frequency response data, or input-output measurements [7]. The latter will be the focus in this thesis.

1.2 Problem Statement

This research will focus on two primary objectives. The first objective is to investigate the existing state-space identification methods that only use input-output sequences, and aim to improve these methods.

The second objective is to develop a method to solve the minimal state space realization problem for MPL systems, where the only observable data from the system are the input-output sequences.

In addition to the two primary objectives, it became evident that the literature lacks

a clearly defined method for modeling capacity constraints in MPL systems. As a result, an extra objective is to design a method to model capacity constraints in MPL systems. Capacity constraints will be clearly defined in Section 3.1.

1.3 Outline

In Chapter 2, we will discuss some preliminaries related to the max-plus algebra and present related work to state space identification and minimal state space realizations for MPL systems. Chapter 3 presents a method for modeling capacity constraints in an MPL system, and we will show how to model the max-plus state space identification problem as a mixed-integer program (MIP) and linear program (LP). In Chapter 4, we will show some numerical experiments in which we compare the computation times of a mixed-integer quadratic program (MIQP), MIP, and an LP for different model dimensions. We will first discuss the experimental setup and afterwards show the results. Chapter 5 is specifically focused on the minimal state space realization problem. In this chapter we will present an algorithm to solve this problem. The thesis ends with some conclusions and recommendations discussed in Chapter 6.

Chapter 2

Preliminaries

In this chapter we will first discuss the basics of the max-plus algebra and give some examples on the max-plus algebra. Subsequently, we will dicuss the related work on state space identification and minimal state space realization of max-plus linear systems.

2.1 The Max-Plus Algebra

This section gives an overview of the theoretical foundations of the max-plus algebra, drawing primarily from the work of [11]. The paper has provided a clear overview of the max-plus algebra. We begin by introducing the key concepts and definitions of the max-plus algebra as presented in the paper. To obtain a more comprehensive understanding of the max-plus algebra, interested readers are referred to [1] and [5]. In addition, we give an example of how to model a system as an MPL system and explain some methods to visualise these systems.

2.1.1 Basic Operations of Max-Plus Algebra

As the name already suggests, the max-plus algebra consists of the basic operations maximization and addition [1], [5]. These are represented by $\oplus$ and $\otimes$ respectively:

$$x \oplus y = \max(x, y) \qquad \text{and} \qquad x \otimes y = x + y$$

for $x, y \in \mathbb{R}_\varepsilon \stackrel{\text{def}}{=} \mathbb{R} \cup \{-\infty\}$.

These symbols are used since there is a remarkable analogy between $\oplus$ and $\otimes$, and the conventional addition and multiplication, respectively. This association enables a substantial number of properties and concepts from linear algebra to be transposed

onto the max-plus algebra by substituting $+$ by $\oplus$ and $\times$ by $\otimes$ [1], [5].

It is worth noting that there exist certain differences between the conventional algebra and max-plus algebra. A primary distinction is the absence of inverse elements with respect to $\oplus$ in $\mathbb{R}_\varepsilon$ in general. In addition, the zero element for $\oplus$ in max-plus algebra is denoted by $\varepsilon \stackrel{\text{def}}{=} -\infty$, and the identity element is denoted by $e \stackrel{\text{def}}{=} 0$.

In the max-plus algebra, the order of evaluation corresponds to that of conventional algebra. Specifically, the operator $\otimes$ takes precedence over $\oplus$.

For all $a \in \mathbb{R}_\varepsilon$, exponentiation of the max-plus algebra is defined as:

$$a^{\otimes^n} = \underbrace{a \otimes a \otimes ... \otimes a}_{n \text{ times}} = \underbrace{a + a + ... + a}_{n \text{ times}} = na$$

The max-plus algebra satisfies the following properties:

1. **Associativity**. For all $a, b, c \in \mathbb{R}_\varepsilon$, we have $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ and $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.
2. **Commutativity**. For all $a, b \in \mathbb{R}_\varepsilon$, we have $a \oplus b = b \oplus a$ and $a \otimes b = b \otimes a$.
3. **Distributivity** of $\otimes$ over $\oplus$. For all $a, b, c \in \mathbb{R}_\varepsilon$, we have $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ and $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$.
4. **Idempotency** of $\oplus$. For all $a \in \mathbb{R}_\varepsilon$, we have $a \oplus a = a$.
5. **Zero element**. For all $a \in \mathbb{R}_\varepsilon$, we have $a \oplus \varepsilon = \varepsilon \oplus a = a$.
6. **Absorbing zero element**. For all $a \in \mathbb{R}_\varepsilon$, we have $a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$.
7. **Identity element**. For all $a \in \mathbb{R}_\varepsilon$, we have $a \otimes e = e \otimes a = a$.

2.1.2 Matrix Operations in Max-Plus Algebra

The basic max-plus operations $\oplus$ and $\otimes$ can be extended to matrices. Suppose we have matrices $A, B \in \mathbb{R}_\varepsilon^{m \times n}$ and $C \in \mathbb{R}_\varepsilon^{n \times p}$ then for all i, j we have

$$(A \oplus B)_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij})$$
$$(A \otimes C)_{ij} = \bigoplus_{k=1}^{n} a_{ik} \otimes c_{kj} = \max_k (a_{ik} + c_{kj})$$

The zero matrix in max-plus algebra is defined as $\mathcal{E}_{m \times n} \in \mathbb{R}_\varepsilon^{m \times n}$ where $(\mathcal{E}_{m \times n})_{ij} = \varepsilon$ for all i, j. The identity matrix is defined as E_n where $(E_n)_{ii} = e$ for all $i \in \{1, ..., n\}$

and $(E_n)_{ij} = \varepsilon$ for all i, j where $i \neq j$. If we would multiply the identity matrix E_n with any vector $x \in \mathbb{R}_\varepsilon^n$, we would see that $E_n \otimes x = x$.

For matrix exponentiation, we have $A^{\otimes^0} = E_n$ and

$$A^{\otimes^n} = \underbrace{A \otimes A \otimes ... \otimes A}_{n \text{ times}}.$$

Example 2.1.2.1. Given $\alpha = 2, A = \begin{pmatrix} e & 1 \\ \varepsilon & 4 \end{pmatrix}$ and $B = \begin{pmatrix} -2 & 4 \\ 1 & \varepsilon \end{pmatrix}$, the following computations can be done:

$$A \oplus B = \begin{pmatrix} \max(e, -2) & \max(1, 4) \\ \max(\varepsilon, 1) & \max(4, \varepsilon) \end{pmatrix} = \begin{pmatrix} e & 4 \\ 1 & 4 \end{pmatrix}$$

$$\alpha \otimes A = \begin{pmatrix} 2 \otimes e & 2 \otimes 1 \\ 2 \otimes \varepsilon & 2 \otimes 4 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ \varepsilon & 6 \end{pmatrix}$$

$$A \otimes B = \begin{pmatrix} \max(e-2, 1+1) & \max(e+4, 1+\varepsilon) \\ \max(\varepsilon-2, 4+1) & \max(\varepsilon+4, 4+\varepsilon) \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 5 & \varepsilon \end{pmatrix}$$

$$B \otimes A = \begin{pmatrix} \max(-2+e, 4+\varepsilon) & \max(-2+1, 4+4) \\ \max(1+e, \varepsilon+\varepsilon) & \max(1+1, \varepsilon+4) \end{pmatrix} = \begin{pmatrix} -2 & 8 \\ 1 & 2 \end{pmatrix}$$

$$A^{\otimes^2} = A \otimes A = \begin{pmatrix} \max(e+e, 1+\varepsilon) & \max(e+1, 1+4) \\ \max(\varepsilon+e, 4+\varepsilon) & \max(\varepsilon+1, 4+4) \end{pmatrix} = \begin{pmatrix} e & 5 \\ \varepsilon & 8 \end{pmatrix}$$

2.1.3 Max-Plus Linear Systems

Discrete event systems (DES) with only synchronization and no concurrency can be modeled by a max-plus-algebraic model. Synchronization requires the availability of several resources or users at the same time, whereas concurrency appears, for instance, when some users must choose among several resources, at a certain time [1]. In this research, we assume that each item flowing through the system has a fixed route. As a result, there is no concurrency.

The max-plus algebraic model is of the following form:

$$\begin{aligned} x(k) &= A \otimes x(k-1) \oplus B \otimes u(k) \\ y(k) &= C \otimes x(k) \end{aligned} \tag{2.1}$$

with $A \in \mathbb{R}_\varepsilon^{n \times n}$, $B \in \mathbb{R}_\varepsilon^{n \times m}$ and $C \in \mathbb{R}_\varepsilon^{p \times n}$, where m is the number of inputs and p the number of outputs. The vector x represents the state, u is the input vector, and y is the output vector of the system. It should be noted that u, x, and y are event times and that k is an event counter. For all $k \in \{1, ..., N\}$, we have $x(k) \in \mathbb{R}_\varepsilon^{n \times 1}$, $y(k) \in \mathbb{R}_\varepsilon^{p \times 1}$, and $u(k) \in \mathbb{R}_\varepsilon^{m \times 1}$, where N is the total number of events.

Example of a simple system

To clarify, suppose that we have a system as in Figure 2.1. The system consists of three resources each with different processing times T_i for $i \in \{1, 2, 3\}$. Raw materials are fed to resources 1 and 2, and subsequently processed by these resource. When resources 1 and 2 are finished processing, resource 3 can start processing. After resource 3 is finished, the item leaves the system. In this example, it is assumed that the time to transport items is negligible and that each resource starts working as soon as possible. Furthermore, the buffers are initially empty and none of the resources are stocked with raw material or intermediate items. We define the event times as follows:

- $u(k)$: time instant at which raw material is fed to the system for the k^{th} time.
- $x_i(k)$: time instant at which the i^{th} resource starts processing for the k^{th} time.
- $y(k)$: time instant at which the k^{th} finished item leaves the system.

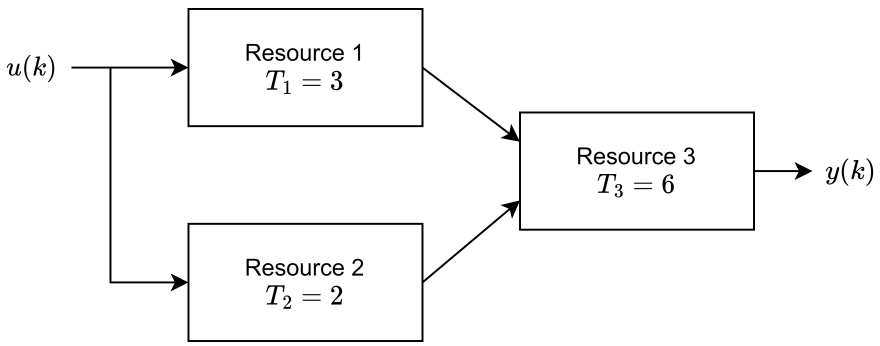


Figure 2.1: A simple system.

Before resource 1 can start processing for the k^{th} time, it should first of all be finished with the previous item. Resource 1 started processing the $(k-1)^{\text{st}}$ part at $x_1(k-1)$. Since it takes $T_1 = 3$ time units to process the raw material, we know that $x_1(k) \geq x_1(k-1) + 3$. In addition, the k^{th} raw material should have reached resource 1. As a result, we know that $x_1(k) \geq u(k)$. Because of the assumption that resources start processing as soon as possible, we have

$$x_1(k) = \max(x_1(k-1) + 3, u(k))$$

for $k \in \{1, ..., N\}$ with N being the total number of items passing through the system.

Using a similar reasoning, we find the following expressions:

$$\begin{aligned} x_2(k) &= \max(x_2(k-1)+2, u(k)) \\ x_3(k) &= \max(x_3(k-1)+6, x_1(k)+3, x_2(k)+2) \\ &= \max(x_3(k-1)+6, x_1(k-1)+6, u(k)+3, x_2(k-1)+4, u(k)+2) \\ &= \max(x_3(k-1)+6, x_1(k-1)+6, u(k)+3, x_2(k-1)+4) \\ y(k) &= x_3(k)+6 \end{aligned}$$

for $k \in \{1, ..., N\}$. We have the initial condition $x_i(0) = \varepsilon$ for $i \in \{1, 2, 3\}$, which corresponds to the assumption that initially buffers are empty and none of the resources are stocked with raw material or intermediate items.

We can now rewrite the max-plus equations above in matrix notation. The system will look as follows:

$$\begin{aligned} x(k) &= \begin{pmatrix} 3 & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon \\ 6 & 4 & 6 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} e \\ e \\ 3 \end{pmatrix} \otimes u(k) \\ y(k) &= \begin{pmatrix} \varepsilon & \varepsilon & 6 \end{pmatrix} \otimes x(k) \end{aligned}$$

where $x(k) = \begin{pmatrix} x_1(k) & x_2(k) & x_3(k) \end{pmatrix}^T$. The model is now of the form as in (2.1). It has to be noted that this is the *explicit model* of the system. In the *implicit model* the vector $x(k)$ is also dependent on $x(k)$. For this example the implicit model looks as follows:

$$\begin{aligned} x(k) &= \begin{pmatrix} e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon \\ 3 & 2 & e \end{pmatrix} \otimes x(k) \oplus \begin{pmatrix} 3 & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & 6 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} e \\ e \\ 3 \end{pmatrix} \otimes u(k) \\ y(k) &= \begin{pmatrix} \varepsilon & \varepsilon & 6 \end{pmatrix} \otimes x(k) \end{aligned}$$

In this thesis, our general assumption is that we aim to model the explicit form.

Dependence on previous cycles

In certain scenarios, the occurrence of events do not only depend on the previous cycle, but also cycles before that. In other words, the event is dependent on events in cycle $k - c$, where $c \in \{1, ..., k\}$. Consequently, the state equation can be reformulated as

$$x(k) = \bigoplus_{c=1}^{k} A_c \otimes x(k-c) \oplus B \otimes u(k)$$

where each matrix $A_c \in \mathbb{R}_{\varepsilon}^{n\times n}$. If $x(k)$ does not depend on $x(k-c)$ for some $c \in \{1, ..., k\}$, that part could be removed from the equation, or the matrix A_c could be set equal to the zero matrix $\mathcal{E}_{n\times n}$.

2.1.4 Visualizing Max-Plus Linear Systems with Graphs

There exists a close relation between the max-plus algebra and graph theory. In particular, there is a subclass of timed Petri nets, called timed event graphs (TEGs), that have been identified to capture the class of stationary MPL systems [4]. In TEGs each place has exactly one input and one output transition and all arcs have weight 1. A TEG consists of a set of places $P = \{p_1, ..., p_n\}$, a set of transitions $T = \{t_1, ..., t_m\}$ and a set of arcs $A \subseteq (P \times T) \cup (T \times P)$. In Figure 2.2, an example of a TEG can be seen. This is the TEG for the system depicted in Figure 2.1. In a TEG, each place is represented by a circle and each transition by a rectangle. As can be seen, each place has exactly one input transition (incoming arc from a rectangle) and one output transition (outgoing arc to a rectangle). The numerical value next to a place denotes the sojourn time, representing the number of time units required to elapse before the token becomes available for the firing of the output transition. In a manufacturing system, this will generally be equal to the processing time of a resource.

Timed event graphs provide a formal and structured way to model, analyze, and reason about temporal relationships between events. Furthermore, TEGs are valuable for visualising the structure of MPL systems and it is straightforward to construct an MPL system from a TEG. As an example, if we define $x_1(k)$ to be the k^{th} firing of transition x_1, then we can easily determine from the TEG how to calculate $x_1(k)$. As can be seen in Figure 2.2, x_1 exhibits two incoming arcs: one through u and the other via the previous firing of x_1. Since a transition is fired as soon as there is an available token in each input place, we can construct the equation $x_1(k) = \max(u(k) + 0, x_1(k) + 3)$. This is the same equation as we had discovered before in our example. If the TEG correctly represents a system, it allows for a more systematic approach in creating the max-plus equations by taking into account all incoming arcs of a transition.

Another method to visualise the constraints of an MPL system is to create a constraint graph. This is a little bit more intuitive than the TEG representation. It can actually be done by unrolling the timed event graph for each firing. To clarify, first we define $u(k)$, $x_i(k)$ and $y(k)$ the same as in Section 2.1.3. In Figure 2.3, the constraint graph for the system of Figure 2.1 can be seen. In the constraint graph, each

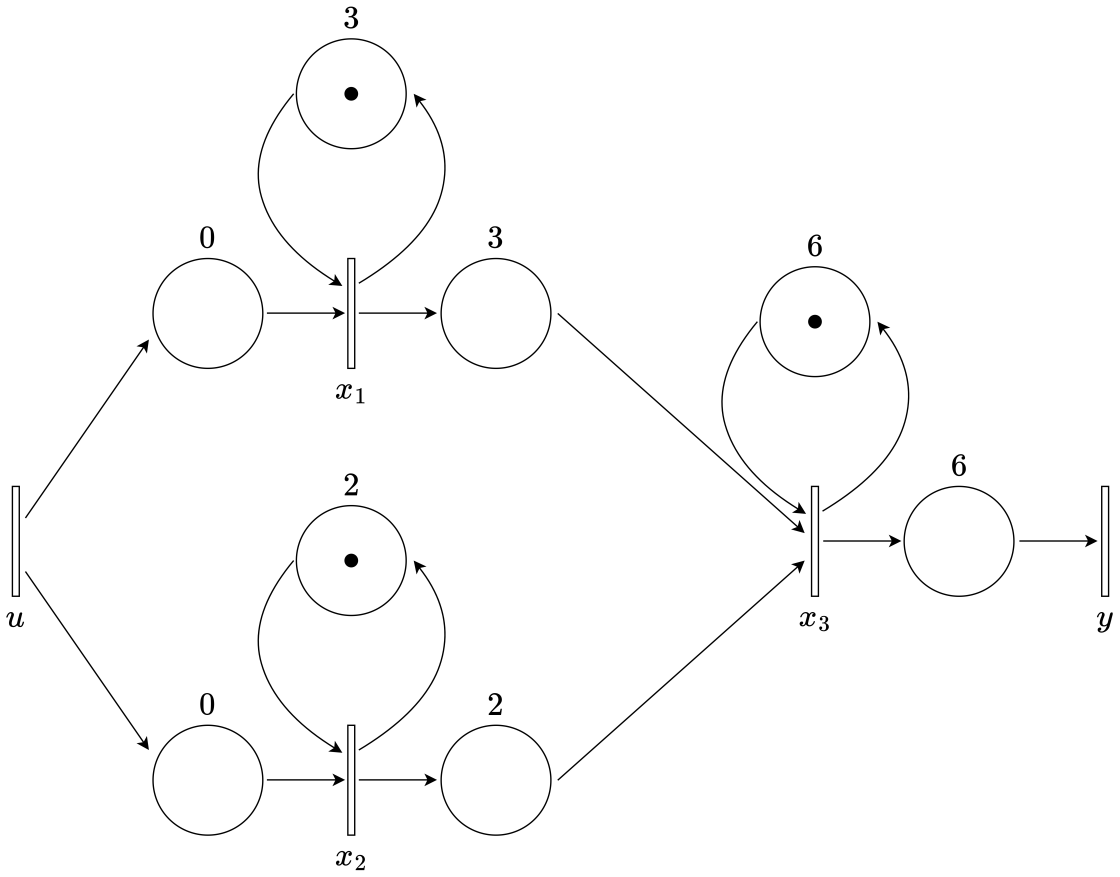


Figure 2.2: Timed event graph of the system in Figure 2.1.

node fires as soon as all incoming arcs are ready. As an example, $x_1(2)$ starts as soon as $u(2)+0$ and $x_1(1)+3$ are satisfied. This can be translated to the constraint $x_1(2) = \max(u(2)+0, x_1(1)+3)$. From this graph it is immediately clear that we can do this for all $k \in \{1, ..., N\}$, resulting in the equation $x_1(k) = \max(u(k), x_1(k-1)+3)$. Because of the assumption that $x_i(0) = \varepsilon$ for all $i \in \{1, 2, 3\}$, we have omitted the arcs from $x_i(0)$ to $x_i(1)$. Again, the max-plus equations can be easily constructed from this graph.

2.2 Related Work

In the early 1960s, Cuninghame-Green [6], and Giffler [3], made independent discoveries that max-linear models could be used to describe certain classes of discrete event systems (DES). Later on Cuninghame-Green extended his work on max-algebraic system theory by publishing [5]. Together with the book of Baccelli et al. [1], he layed the foundations of the max-plus algebra and the MPL system theory.

In this section, we will discuss the related work to state space identification of MPL

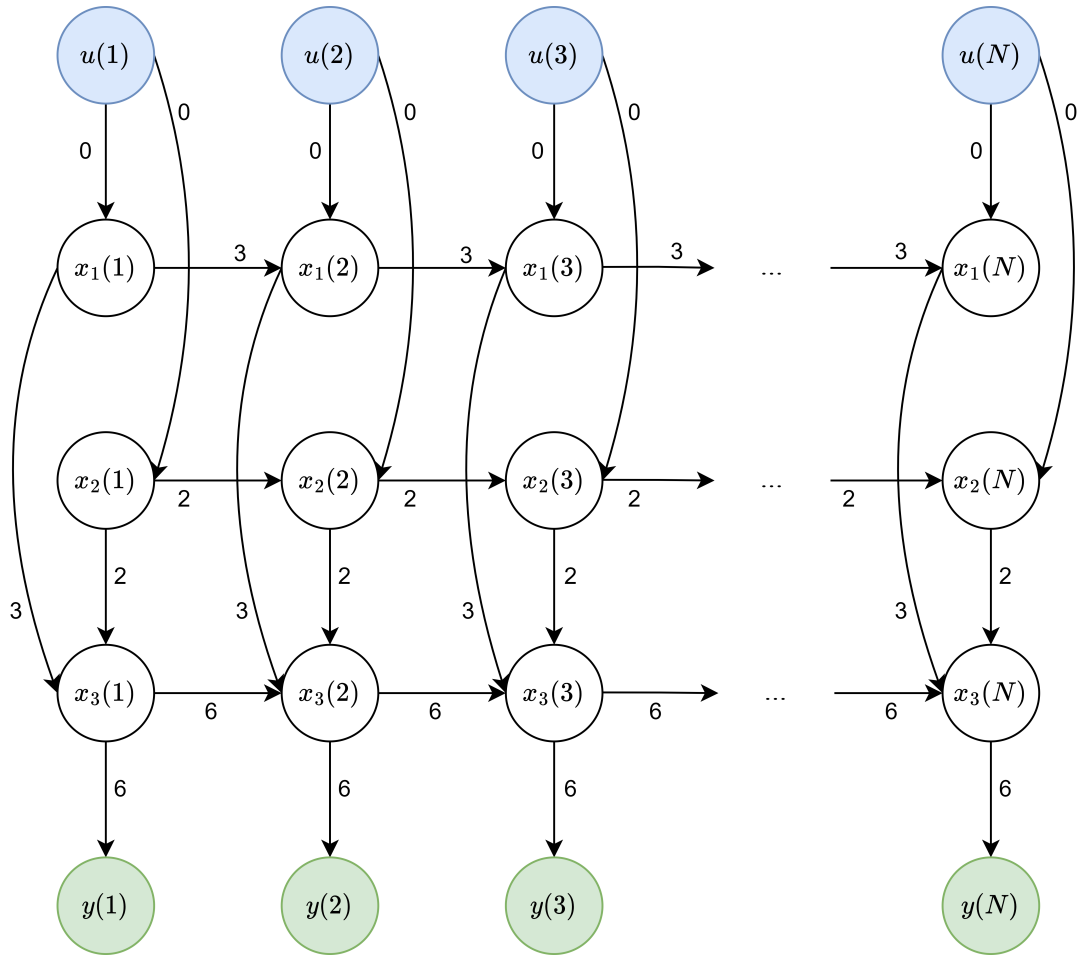


Figure 2.3: Constraint graph of the system in Figure 2.1.

systems and subsequently discuss related work to minimal state space realization methods in the max-plus algebra.

2.2.1 State Space Identification

In the 1990s, Boimond, Hardouin, Chiron, and Gallot [2], [15] were one of the first researchers investigating the identification problem for MPL discrete event systems. In their research, they used a transfer function approach to estimate the parameters of MPL systems. As in conventional system theory, they transformed the transfer function into an ARMA model to obtain a finite number of parameters. Afterwards, they defined a prediction error and minimized it using residuation theory. The authors mentioned that they could not guarantee the minimality of the model due to the considered expression of the transfer function.

As was mentioned by De Schutter et al. [12], the connection with the physical structure is usually lost when transforming an identified transfer function into a state

space model, whereas the state space matrices of an MPL system clearly represent the physical layout of the system. In addition, the extension from single-input, single-output (SISO) to multiple-input, multiple-output (MIMO) is more intuitive for state space models, and they have the capability to uncover hidden behaviour, such as unobservable modes. Therefore, we focus on state space identification in this thesis.

In Schullerus and Krebs [27], a state space identification method has been derived. However, in their paper the state is assumed to be measurable and the internal structure of the system is assumed to be completely known. In this thesis, this is not the case. We assume that only input-output sequences can be observed. This is in line with the assumptions of the paper from De Schutter et al. [12]. In their paper, they transformed the state space identification problem into a mixed integer quadratic programming (MIQP) problem. Within their MIQP they try to minimize the error between the estimated output times and actual output times given the constraints of the MPL system. They include measurement noise. In this thesis, we will show that, if we assume there will be no measurement noise or noise on the processing times as in [15], [25], [28], we can reformulate this problem into a mixed integer program (MIP), and in special cases even be able to solve this problem using a linear program (LP).

It should be noted that some other methods have been proposed for the state space identification of MPL systems as well. Menguy et al. [25] developed a method to estimate the parameters using an impulse response, and in De Schutter and De Moor [9] the authors use the system's Markov parameters to find the system parameters. Furthermore, Hook [22] presented new theory and algorithms for 2-norm regression over the max-plus semiring and showed how this could be used in system identification. He assumed that the states were measurable. As already mentioned, in this thesis we focus mainly on the methods discussed in De Schutter et al. [12]. In Section 3.2.2, we will describe their problem and method in more detail.

Furthermore, some methods have been proposed for the state identification problem. This problem concerns the correct estimation of the state x whenever there is some uncontrollable noise involved. In this case the parameters of the matrices $A, B,$ and C are known. Hardouin et al. [18] proposed a method to find the state estimates based on the Luenberger observer in classical linear systems theory, and Lai et al. [23] proposed an algorithm based on the fact that the dynamic behaviour of a max-plus automaton can be characterized by its state vector. It has to be noted that this problem is related to the problem in this thesis, but concerns different as-

sumptions. In addition to finding estimates of the system matrices A, B, and C, our method will also give estimates on the state of the system.

It is important to emphasize that this thesis is focused on deterministic MPL systems. Nonetheless, research has also been conducted on identification of stochastic MPL systems, as explored by Farahani et al. [14] and Van den Boom et al. [34]. These authors have proposed different methods to solve this problem.

In addition to estimating the parameters of an MPL system, It is valuable to be aware that the input signal design has an influence on the identification of the matrices. If the input signal design is chosen incorrectly, different modes of MPL systems could be unobservable and thus not be identified. In Schullerus et al. [29], the authors propose a new method for the input signal design. The input signal design will not be the focus of this thesis.

2.2.2 Minimal State Space Realization for Max-Plus Linear Systems

Before diving into the theory, it is worth mentioning that there is a small difference between minimal realization and model reduction. Model reduction is focused on reducing the complexity of an already existing high-order model, while minimal realization is focused on finding a model of the minimal order representation of a system given some data. Examples of model reduction are balanced truncation [26], Krylov subspace methods [20], and singular value decomposition.

The minimal realization problem could be solved using a whole spectrum of different methods, including methods using transfer functions and state space models. In this thesis, the focus will be on state space models. Gilbert [16] first introduced the minimal state space realization problem for linear time-invariant systems. In his work, Gilbert proposed a method aimed at converting a transfer function into a set of differential equations. After Gilbert's work, multiple methods have been proposed to solve this problem for linear systems, such as Silverman's algorithm [31] and Ho's algorithm [21]. However, despite the remarkable analogies between the max-plus algebra and linear algebra, the process of translating properties and algorithms from linear algebra to the max-plus algebra is not always straightforward. As was mentioned by De Schutter and De Moor [10], the minimal state space realization problem is one of the problems that is easily solved in linear system theory, but a complex problem in max-algebraic system theory.

In 1995, De Schutter and De Moor [9] showed that the minimal state space realization problem in the max-plus algebra could be formulated as an extended linear complementarity problem (ELCP), and solved this problem for the first time. As was shown by the same authors, the general ELCP is an NP-hard problem [8]. In De Schutter [7], it is even stated that there are strong indications that the general max-plus-algebraic minimal state space realization problem is at least NP-hard. Not a lot of methods have been proposed in literature to solve the minimal state space realization problem for MPL systems. In 1997, De Schutter and De Moor [10] proposed another method in which they used a matrix factorization algorithm to determine a lower bound for the minimal system order. Their procedure could often be used to compute the system matrices of a minimal state space realization as well. However, in both of their papers from 1995 and 1997 they assumed that the known data consisted of impulse responses, while the focus in this thesis will be on input-output sequences. We will formulate a method to model the minimal state space realization problem in the max-plus algebra as a mix of multiple MIPs.

Chapter 3

Modeling

This chapter is focused on the modeling part of this thesis. We will first show how capacity constraints can be modelled in MPL systems. Afterwards, we will explore methods for representing the max-plus identification problem as an MIP problem and give a method to solve the problem using an LP for certain cases.

3.1 Modeling Capacity Constraints

As was mentioned in Section 1.2, one of the goals of this research was to design a method to model capacity constraints in MPL systems. Capacity constraints are constraints on one or multiple resources, where the number of items in those resources cannot exceed a certain maximum capacity. We will first show an example of how to model a capacity constraint in an MPL system, and afterwards give a method for generating the max-plus equations for systems with capacity constraints.

3.1.1 Example of Modeling Capacity Constraints

Suppose we have a system as in Figure 3.1. This system consists of two resources with a processing time of $T_1 = 1$ and $T_2 = 3$. Both resources can process 1 item at a time, indicated by the maximum capacity a_1 and a_2 of 1. Furthermore, this system has a capacity constraint over both resources 1 and 2 of $c = 2$, indicating that at most 2 item can be in the system at the same time.

This capacity constraint creates an extra constraint on both $x_1(k)$ and $x_2(k)$. To get a better understanding, let's first look at the constraint graph of this system, depicted in Figure 3.2. We assume that $x_i(k) = \varepsilon$ for $k \leq 0$ for $i \in \{1, 2\}$. For convenience the arcs coming from $x_i(k)$ for $k \leq 0$ are neglected. We see that each $x_1(k)$ has three incoming arcs, one from $u(k)$, one from $x_1(k-1)$, and one from $x_2(k-2)$. For $x_2(k)$,

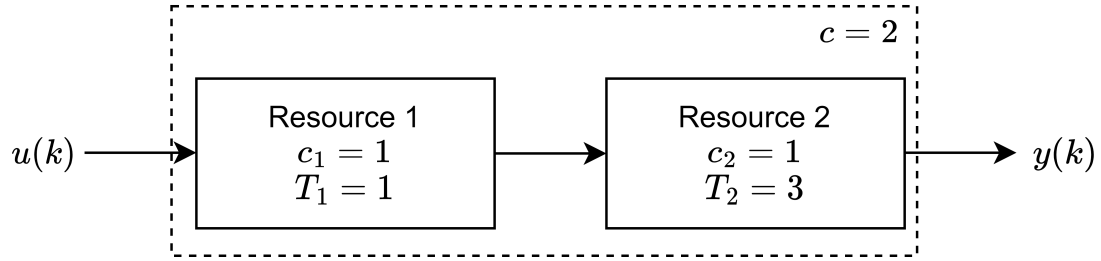


Figure 3.1: Example of a system with a capacity constraint.

we see that it has two incoming arcs, one from $x_1(k)$, and the other from $x_2(k-1)$. The resulting max-plus model for this system is as follows:

$$x(k) = \begin{pmatrix} T_1 & \varepsilon \\ T_1 + T_1 & T_2 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} \varepsilon & T_2 \\ \varepsilon & T_1 + T_2 \end{pmatrix} \otimes x(k-2) \oplus \begin{pmatrix} e \\ T_1 \end{pmatrix} \otimes u(k)$$
$$y(k) = \begin{pmatrix} \varepsilon & T_2 \end{pmatrix} \otimes x(k)$$

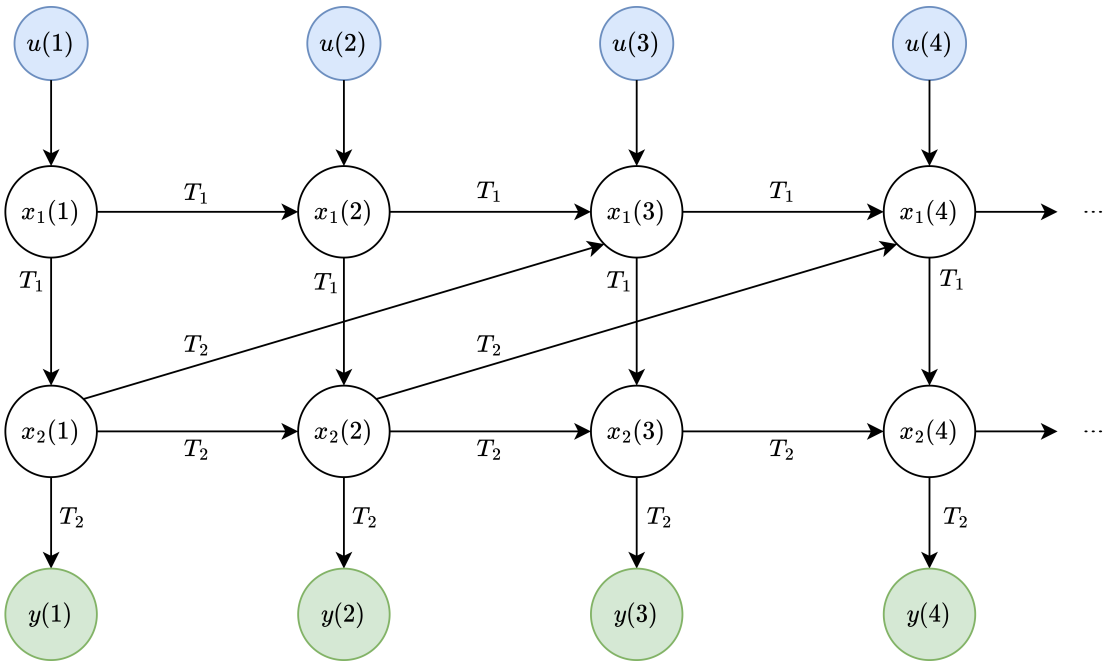


Figure 3.2: Constraint graph of the system of Figure 3.1. For convenience the arcs coming from $x_i(k)$ for $k \leq 0$ for $i \in \{1, 2\}$ are neglected.

As can be seen in the previous example, the extra matrix used to model the capacity constraint is the matrix $A_2 = \begin{pmatrix} \varepsilon & T_2 \\ \varepsilon & T_1 + T_2 \end{pmatrix}$, which is multiplied by the vector $x(k-2)$.

3.1.2 Generalization of Modeling Capacity Constraints

In the previous section, we have shown an example of how to model a capacity constraint in a simple system. In this section, we will establish a general method of how to model a capacity constraint. Before we can do this, we first need to introduce the term *capacity restriction area*. A capacity restriction area is defined as the area where the capacity constraint is active. We make the assumption that there is a single location where an item can leave the capacity constraint. For most systems this makes sense. However, if this is not the case, other methods have to be developed. In the example of Section 3.1.1, the capacity restriction area includes both resources 1 and 2.

Suppose an item has a certain path through a system, where it enters a capacity restriction area at resource i and leaves it after resource j. Suppose the maximum capacity is c. Then the starting time of event $x_j(k-c)$ will have an influence on all resources in the path after and including resource i. Suppose, without loss of generality, that the path through the system passes the following resources:

$$(1 \to ... \to i-1 \to i \to i+1 \to ... \to j-1 \to j \to j+1 \to ... \to n)$$

where n is the last resource before leaving the system. Let us denote T_x as the processing time of resource x. Resource x for $x \in \{i, i+1, ..., n\}$ can only start processing the k^{th} item when at least $T_j + T_i + T_{i+1} + ... + T_{x-1}$ time units have passed after the event time $x_j(k-c)$. As a result, we could construct the following matrix:

$$\mathsf{A}_c = \begin{pmatrix} a_{1,1} & \cdots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \cdots & a_{n,1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & & a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} & \cdots & \vdots \\ \vdots & & a_{i,j-1} & a_{i,j} & a_{i,j+1} & \cdots & \vdots \\ \vdots & & a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} & \cdots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,j-1} & a_{n,j} & a_{n,j+1} & \cdots & a_{n,n} \end{pmatrix} = \begin{pmatrix} \epsilon & \cdots & \epsilon & \epsilon & \epsilon & \cdots & \epsilon \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & & \epsilon & \epsilon & \epsilon & \cdots & \vdots \\ \vdots & & \epsilon & T_y & \epsilon & \cdots & \vdots \\ \vdots & & \epsilon & T_xT_y & \epsilon & \cdots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ \epsilon & \cdots & \epsilon & T_xT_{x+1}...T_{n-1}T_y & \epsilon & \cdots & \epsilon \end{pmatrix}$$

To prevent the matrix above from becoming excessively large, the $\otimes$ operation has been omitted between the processing times. As an example, $a_{i+1,j} = T_xT_y = T_x \otimes T_y = T_x + T_y$. The given matrix can be effectively used within an MPL system with n-serial resources. As an example, let us consider the system depicted in Figure 3.3. Utilizing the structure of the matrix above, generating the max-plus equations becomes straightforward. The resulting max-plus equations can be expressed as follows:

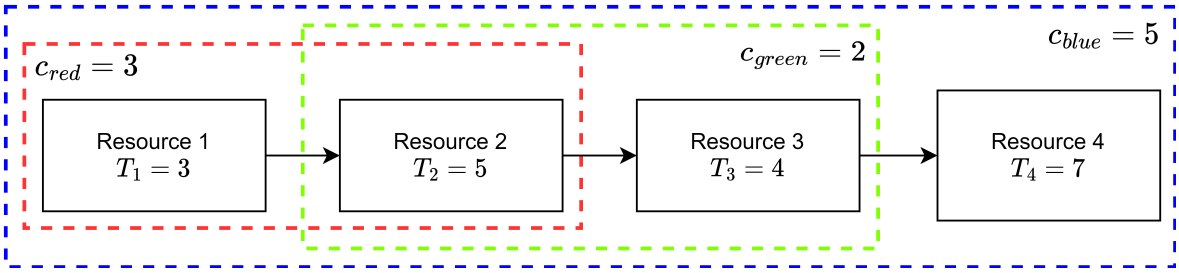


Figure 3.3: System with 4-serial resources with three capacity constraints.

$$x(k) = \begin{pmatrix} 3 & \varepsilon & \varepsilon & \varepsilon \\ 6 & 5 & \varepsilon & \varepsilon \\ 11 & 10 & 4 & \varepsilon \\ 15 & 14 & 8 & 7 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 4 & \varepsilon \\ \varepsilon & \varepsilon & 9 & \varepsilon \\ \varepsilon & \varepsilon & 13 & \varepsilon \end{pmatrix} \otimes x(k-2) \oplus$$

$$\begin{pmatrix} \varepsilon & 5 & \varepsilon & \varepsilon \\ \varepsilon & 8 & \varepsilon & \varepsilon \\ \varepsilon & 13 & \varepsilon & \varepsilon \\ \varepsilon & 17 & \varepsilon & \varepsilon \end{pmatrix} \otimes x(k-3) \oplus \begin{pmatrix} 7 & \varepsilon & \varepsilon & \varepsilon \\ 10 & \varepsilon & \varepsilon & \varepsilon \\ 15 & \varepsilon & \varepsilon & \varepsilon \\ 19 & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} \otimes x(k-5)$$

$$y(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 7 \end{pmatrix} \otimes x(k)$$

The construction of the matrix that is multiplied by $x(k-1)$ is straightforward. An efficient method has been described by Seleim and El Maraghy [30]. In their paper they basically describe that the matrix that will be multiplied by $x(k-1)$ in a system with n-serial resources has the following structure:

$$\begin{pmatrix} T_1 & \epsilon & \dots & \dots & \epsilon \\ T_1^2 & T_2 & \ddots & & \epsilon \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ T_1^2 T_2 ... T_{n-2} & T_2^2 T_3 ... T_{n-2} & \dots & T_{n-1} & \epsilon \\ T_1^2 T_2 ... T_{n-1} & T_2^2 T_3 ... T_{n-1} & \dots & T_{n-1}^2 & T_n \end{pmatrix} \tag{3.1}$$

If there will be multiple capacity constraints with the same maximum capacity, the matrices can of course be combined into one matrix by using the $\oplus$ operation. If for example, the red capacity restriction area of Figure 3.3 had a maximum capacity of 2 ($c_{red} = 2$), then the matrix created by the red and green capacity constraint could be combined into the following matrix:

$$\begin{pmatrix} \varepsilon & 5 & \varepsilon & \varepsilon \\ \varepsilon & 8 & \varepsilon & \varepsilon \\ \varepsilon & 13 & \varepsilon & \varepsilon \\ \varepsilon & 17 & \varepsilon & \varepsilon \end{pmatrix} \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 4 & \varepsilon \\ \varepsilon & \varepsilon & 9 & \varepsilon \\ \varepsilon & \varepsilon & 13 & \varepsilon \end{pmatrix} \oplus = \begin{pmatrix} \varepsilon & 5 & \varepsilon & \varepsilon \\ \varepsilon & 8 & 4 & \varepsilon \\ \varepsilon & 13 & 9 & \varepsilon \\ \varepsilon & 17 & 13 & \varepsilon \end{pmatrix}$$

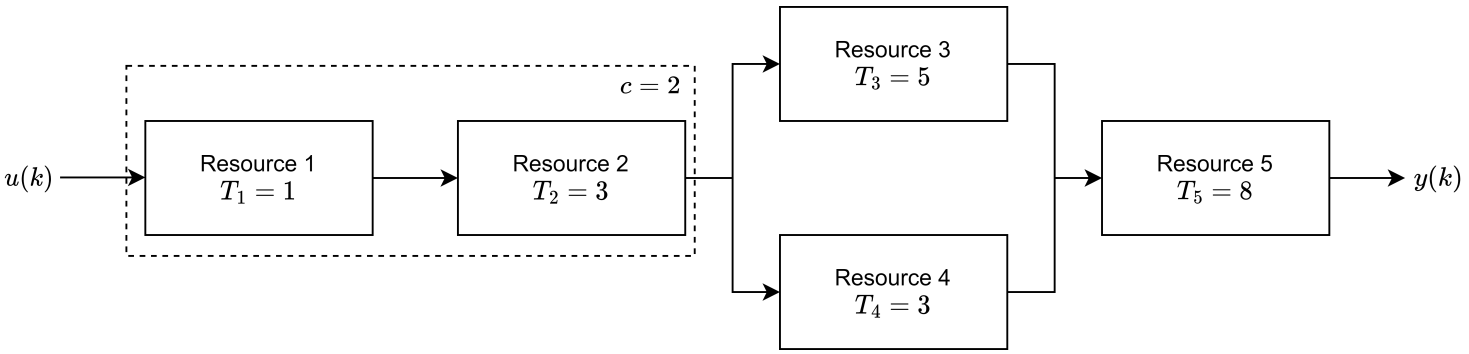


Figure 3.4: System with a capacity constraint and branching.

It is worth noting that in case there exist any transportation times between the resources, they must be accounted for and added to the calculations. This step is relatively straightforward.

With the process that we have described above, it is easy to create the max-plus equations for systems with n-serial resources containing any number of capacity constraints. However, if there are any branches within the system where the process splits from one resource to two or more other resources, there are multiple paths until the item leaves the system. In this particular scenario, the structure of the matrix is slightly different. However, the underlying principle remains unchanged. We can still create a matrix for each path that the system follows. Afterwards, we can combine the different matrices and only include the highest values by using the $\oplus$ operation between the matrices. This holds for any number of matrices. To clarify, suppose we have a system as in Figure 3.4. For the capacity constraint, we can create matrices for both the different paths via resource 3 and resource 4. As already mentioned, this can be done in a similar way as for the system with n-serial resources. However, if we choose the path via resource 3, we have to leave out resource 4, and vice versa. The matrix for the path via resources 3 and 4 will then look as follows, respectively:

$$A_{2,\text{resource 3}} = \begin{pmatrix} \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 7 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 13 & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}, \quad A_{2,\text{resource 4}} = \begin{pmatrix} \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 7 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 10 & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}$$

Subsequently, these matrices can be combined into the matrix A_2, which will be

multiplied by $x(k-2)$ in the MPL system. This matrix looks as follows:

$$A_2 = A_{2,\text{resource 3}} \oplus A_{2,\text{resource 4}} = \begin{pmatrix} \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 7 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 7 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 13 & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}$$

In general, we can state that for any path through the system, and capacity restriction starting at resource i and finishing at resource j with a maximum capacity of c, the event $x_l(k)$ with l being any resource in the path after and including resource i, will occur at least when time $x_j(k-c) + T_j + T_i + T_{i+1} + ... + T_{l-1}$ has passed.

3.2 Mixed-Integer Programming Formulation

In this section, we will dive deeper into the method of how to model the state space identification problem as an MIP. We will first show methods how to model maximization constraints, and afterwards show how to model an MPL system as an MIP. As mentioned before, we assume that the processing times and measurements of $u(k)$ and $y(k)$ in the MPL system are not corrupted by noise.

3.2.1 Modeling Maximization Constraints in an MIP

To effectively model an MPL system as an MIP problem, it is essential to understand how to model a maximization constraint as an MIP constraint. Suppose we have the following maximization function:

$$z = \max(a_1, ..., a_n) \tag{3.2}$$

To model this in an MIP, we can first ensure that $z \geq \max(a_1, ..., a_n)$ by using the linear constraints:

$$z \geq a_i, \qquad \forall i \in \{1, ..., n\} \tag{3.3}$$

In some optimization problems, it can be the case that the objective function enforces that $z \leq \max(a_1, ..., a_n)$ and thus the optimization only requires the constraints as in (3.3). As a result, the constraints do not need any integer variables and we can solve the problem with an LP. However, if this is not the case, it should be enforced by some additional constraints. Let M be a constant upper bound on z, and let $y_i \in \{0, 1\}\ \forall i \in \{1, ..., n\}$ be a binary decision variable. We can now impose the following linear constraints:

$$\sum_{i=1}^{n} y_i \geq 1 \tag{3.4}$$

$$z \leq a_i + M(1 - y_i), \quad \forall i \in \{1, \ldots, n\} \tag{3.5}$$

To proof that these constraints correctly model Equation (3.2), we first show that if Equation (3.2) is true, constraints (3.3), (3.4), and (3.5) are satisfied. Afterwards, we show that if constraints (3.3), (3.4), and (3.5) are satisfied, that Equation (3.2) holds.

First suppose that $z = \max(a_1, ..., a_n)$. This immediately implies that constraint (3.3) holds by definition of the maximization function. We know that there exists an $i \in \{1, ..., n\}$ such that $a_i \geq a_j \ \forall j \in \{1, ..., n\}$. This implies that $a_i = \max(a_1, ..., a_n)$, and thus $z = a_i$. Subsequently, y_i can be set equal to 1 without violating constraint (3.5) and thus constraint (3.4) is also satisfied. For all $j \neq i$, we have that y_j can be set equal to 0, implying that constraint (3.5) is satisfied for all $j \in \{1, ..., n\}$. As a result, (3.2) implies that all constraints (3.3), (3.4), and (3.5) are satisfied.

Now suppose that constraints (3.3), (3.4), and (3.5) are satisfied. This implies that $y_i = 1$ for some $i \in \{1, ..., n\}$. Therefore, by constraint (3.5) we have $z \leq a_i$ and by Constraint (3.3) that $z \geq a_i$. As a result, we have $z = a_i$ for some i. In addition, by constraint (3.3), we see that z is larger than or equal to $a_i \ \forall i \in \{1, ..., n\}$. This implies that $z = \max(a_1, ..., a_n)$ and thus Equation (3.2) holds.

3.2.2 Modeling a Max-Plus Linear System as an MIP

To model an MPL system as an MIP, we can use the logic from Section 3.2.1 to replace all the maximization constraints by using constraints (3.3), (3.4) and (3.5). However, we will first look into the methods that have been used in [12] to create an MIQP. In their paper, they considered the following estimation model of the MPL system:

$$\hat{x}(k) = \hat{A} \otimes \hat{x}(k-1) \oplus \hat{B} \otimes u(k)$$
$$\hat{y}(k) = \hat{C} \otimes \hat{x}(k)$$

where $\hat{A}, \hat{B}, \hat{C}, \hat{x}(k)$ and $\hat{y}(k)$ are the estimates of $A, B, C, x(k)$ and $y(k)$, respectively. In addition, they mention that it is possible to add additional constraints on the estimates if, based on physical constraints or insights, hard upper and lower bounds

are known. The constraints they add are of the form

$$\begin{aligned} A_{\min} \leq \hat{A} \leq A_{\max} \\ B_{\min} \leq \hat{B} \leq B_{\max} \\ C_{\min} \leq \hat{C} \leq C_{\max} \\ x_{\min}(0) \leq \hat{x}(0) \leq x_{\max}(0) \end{aligned}$$

Afterwards, they define the following:

$$\hat{x}_{tot} = \begin{pmatrix} \hat{x}(0) \\ \vdots \\ \hat{x}(N) \end{pmatrix}, \quad \hat{y}_{tot} = \begin{pmatrix} \hat{y}(1) \\ \vdots \\ \hat{y}(N) \end{pmatrix}, \quad y_{tot} = \begin{pmatrix} y(1) \\ \vdots \\ y(N) \end{pmatrix}$$

The goal is to minimize the error between the estimated output $\hat{y}(k)$ and the measured output $y(k)$ for all $k \in \{1, ..., N\}$. They define the MPL state space identification problem as follows:

$$\begin{aligned} &\min_{\hat{A},\hat{B},\hat{C},\hat{x}_{\text{tot}},\hat{y}_{\text{tot}}} && ||\hat{y}_{\text{tot}} - y_{\text{tot}}||_2^2 \\ &\text{subject to} && \hat{x}(k) = \hat{A} \otimes \hat{x}(k-1) \oplus \hat{B} \otimes u(k), \quad k \in \{1, \ldots, N\} \\ & && \hat{y}(k) = \hat{C} \otimes \hat{x}(k), \quad k \in \{1, \ldots, N\} \\ & && A_{\text{min}} \leq \hat{A} \leq A_{\text{max}} \\ & && B_{\text{min}} \leq \hat{B} \leq B_{\text{max}} \\ & && C_{\text{min}} \leq \hat{C} \leq C_{\text{max}} \\ & && x_{\text{min}}(0) \leq \hat{x}(0) \leq x_{\text{max}}(0) \end{aligned} \tag{3.6}$$

To solve the optimization problem, the authors of the paper recast the problem as an Extended Linear Complementarity Problem (ELCP) [8] and use the fact that this problem can be recast as a mixed-integer feasibility problem. Together with the quadratic objective function, the authors create an MIQP to solve the optimization problem. This is very similar to using the constraints described in Section 3.2.1 for creating the maximization constraints.

As was mentioned before, the assumption in De Schutter et al. [12] was that the processing times are corrupted by noise. Due to this assumption, the optimization problem is formulated as an MIQP problem. However, if we assume that the processing times are not corrupted by noise, we can show that the optimization problem can be solved by an MIP, which could be significantly more computationally efficient.

Basically, whenever we assume that there is no noise, we can remove the quadratic objective function and completely remove the variable $\hat{y}(k)$ from the optimization

problem. We could then change the optimization problem into the following:

$$\begin{array}{lll} \text{max} & 0 & \\ \text{subject to} & \hat{x}(k) = \hat{A} \otimes \hat{x}(k-1) \oplus \hat{B} \otimes u(k), & k \in \{1, \ldots, N\} \\ & y(k) = \hat{C} \otimes \hat{x}(k), & k \in \{1, \ldots, N\} \end{array} \tag{3.7}$$

where we have neglected the bounds on $\hat{A}, \hat{B}, \hat{C}$, and $\hat{x}(0)$. These could be added whenever any physical bounds of the system are known. For the rest of the report, we assume that we do not know any bounds on these matrices. However, we do assume in all of our experiments that $\hat{x}(0) = \varepsilon$, since we assume that all initial buffers of the MPL system are empty. Instead of finding the best parameters $\hat{A}, \hat{B}, \hat{C}, \hat{x}_{tot}$, and $\hat{y}_{tot}$ that minimize the sum of squared errors between $\hat{y}_{tot}$ en y_{tot}, the optimization problem is now about finding some feasible solution that perfectly maps the input signal $u(k)$ to the output signal $y(k)$ for all $k \in \{1, ..., N\}$.

Note the difference between the equation $y(k) = \hat{C} \otimes \hat{x}(k)$ of (3.7) and $\hat{y}(k) = \hat{C} \otimes \hat{x}(k)$ of (3.7). It is only a small difference, but it makes a significant difference. Neglecting the variable $\hat{y}(k)$ is possible because there is no noise involved in the process anymore. To clarify this, suppose we have a very simple MPL system consisting of only one resource as in Figure 3.5. If the processing time T_1 is corrupted by noise, the model (3.7) will not be able to find a feasible solution in general. In this simple system, the constraints for (3.7) are basically equal to

$$y(k) = \max(\hat{c}_{1,1} + \hat{a}_{1,1} + \hat{x}(k-1), \hat{c}_{1,1} + \hat{b}_{1,1} + u(k))$$

where $\hat{a}_{1,1}, \hat{b}_{1,1}$, and $\hat{c}_{1,1}$ are the only entries of the matrices $\hat{A}, \hat{B}, \hat{C} \in \mathbb{R}_{\varepsilon}^{1\times 1}$. Since we know exactly how the system looks, we know that $\hat{a}_{1,1} = T_1$, $\hat{b}_{1,1} = 0$, and $\hat{c}_{1,1} = 0$. This means that in the equation above, we have $\hat{c}_{1,1} + \hat{a}_{1,1} = T_1$ and $\hat{c}_{1,1} + \hat{b}_{1,1} = 0$. If we assume that $u(k) = 0$ for all $k \in \{1, ..., N\}$, and that T_1 can be any value between 0.9 and 1.1 time units, this system will obviously not be able to find a feasible solution in general. However, for the model of (3.6) this system can be solved. This is because of the extra decision variable $\hat{y}(k)$ that has been used.

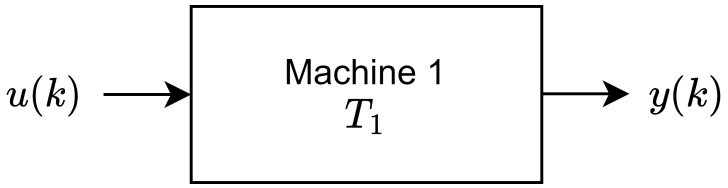


Figure 3.5: System consisting of one resource with a processing time of T_1.

The model (3.6) has N more decision variables than the (3.7), because of the extra $\hat{y}(k)$ that are included for $k \in \{1, ..., N\}$. Again, N here is the total number of items

passing through the system.

The optimization problem of (3.7) can be reformulated into an MIP by replacing the maximization constraints by the constraints (3.3), (3.4), and (3.5) described in Section 3.2.1. In order to do this, we first expand the matrix notation of the optimization problem of (3.7) and arrive at the following notation of the optimization problem:

$$
\begin{array}{lll}
\text{maximize} & 0 & \\
\text{subject to} & \hat{x}_i(k) = \max\{ \; \hat{a}_{i,1} + \hat{x}_1(k-1), \ldots, \hat{a}_{i,n} + \hat{x}_n(k-1), & \\
 & \qquad \hat{b}_{i,1} + u_1(k), \ldots, \hat{b}_{i,m} + u_m(k)\}, & i \in [n], k \in [N] \\
 & y_i(k) = \max\{ \; \hat{c}_{i,1} + \hat{x}_1(k), \ldots, \hat{c}_{i,n} + \hat{x}_n(k)\}, & i \in [p], k \in [N]
\end{array}
\tag{3.8}
$$

where $[n]$ denotes the set $\{1, ..., n\}$. Now we can change the maximization constraints in the optimization problem above into an MIP. The resulting MIP looks as follows:

$$
\begin{array}{lll}
\text{max} & 0 & \\
\text{s.t.} & \hat{x}_i(k) \geq \hat{a}_{i,j} + \hat{x}_j(k-1), & i,j \in [n], k \in [N] \\
 & \hat{x}_i(k) \geq \hat{b}_{i,j} + u_j(k), & i \in [n], j \in [m], k \in [N] \\
 & \hat{x}_i(k) \leq \hat{a}_{i,j} + \hat{x}_j(k-1) + M(1 - d_{i,j}), & i,j \in [n], k \in [N] \\
 & \hat{x}_i(k) \leq \hat{b}_{i,j} + u_j(k) + M(1 - d_{i,j+n}), & i \in [n], j \in [m], k \in [N] \\
 & \sum_{j=1}^{n+m} d_{i,j} \geq 1, & i \in [n] \\
 & y_i(k) \geq \hat{c}_{i,j} + \hat{x}_j(k), & i \in [p], j \in [n], k \in [N] \\
 & y_i(k) \leq \hat{c}_{i,j} + \hat{x}_j(k) + M(1 - e_{i,j}), & i \in [p], j \in [n], k \in [N] \\
 & \sum_{j=1}^{n} e_{i,j} \geq 1, & i \in [n] \\
 & d_{i,j} \in \{0,1\}, & i \in [n], j \in [n+m] \\
 & e_{i,j} \in \{0,1\}, & i \in [p], j \in [n]
\end{array}
\tag{3.9}
$$

It has to be noted that if certain constraints on the system are known or if certain measurements of the state or processing times are available, they could easily be included in the MIP model above.

For the optimization problem of (3.6), the similar method can be used to construct

the following MIQP:

$$
\begin{array}{lll}
\text{max} & ||\hat{y}_{tot} - y_{tot}||_2^2 & \\
\text{s.t.} & \hat{x}_i(k) \geq \hat{a}_{i,j} + \hat{x}_j(k-1), & i,j \in [n], k \in [N] \\
& \hat{x}_i(k) \geq \hat{b}_{i,j} + u_j(k), & i \in [n], j \in [m], k \in [N] \\
& \hat{x}_i(k) \leq \hat{a}_{i,j} + \hat{x}_j(k-1) + M(1-d_{i,j}), & i,j \in [n], k \in [N] \\
& \hat{x}_i(k) \leq \hat{b}_{i,j} + u_j(k) + M(1-d_{i,j+n}), & i \in [n], j \in [m], k \in [N] \\
& \sum_{j=1}^{n+m} d_{i,j} \geq 1, & i \in [n] \\
& \hat{y}_i(k) \geq \hat{c}_{i,j} + \hat{x}_j(k), & i \in [p], j \in [n], k \in [N] \\
& \hat{y}_i(k) \leq \hat{c}_{i,j} + \hat{x}_j(k) + M(1-e_{i,j}), & i \in [p], j \in [n], k \in [N] \\
& \sum_{j=1}^{n} e_{i,j} \geq 1, & i \in [n] \\
& d_{i,j} \in \{0,1\}, & i \in [n], j \in [n+m] \\
& e_{i,j} \in \{0,1\}, & i \in [p], j \in [n]
\end{array}
\tag{3.10}
$$

Now that the MIQP and MIP have been constructed, numerical experiments can be conducted to evaluate the computation time for both models for various MPL systems. Before we focus on that, we first want to describe the possibilities of modeling an MPL system as an LP.

3.2.3 Modeling a Max-Plus Linear System as an LP

As was already mentioned in Section 3.2.1, in some optimization problems, the objective function enforces that the constraints (3.3) are tight for at least one of them. To clarify, suppose that we have the following optimization problem:

$$
\begin{array}{ll}
\text{min} & z \\
\text{s.t.} & z \geq a_i, \quad \forall i \in \{1, ..., n\}
\end{array}
\tag{3.11}
$$

If z is the only decision variable and a_i are given values, we can see that the objective function forces $z = a_i$ for some $i \in \{1, ..., n\}$. This logic could also be used in creating an LP for an MPL system. The LP that we have created looks as follows:

$$
\begin{array}{lll}
\text{max} & \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{a}_{i,j} + \sum_{i=1}^{n} & \sum_{j=1}^{m} \hat{b}_{i,j} + \sum_{i=1}^{p}\sum_{j=1}^{n} \hat{c}_{i,j} \\
\text{s.t.} & \hat{x}_i(k) \geq \hat{a}_{i,j} + \hat{x}_j(k-1), & i,j \in [n], k \in [N] \\
& \hat{x}_i(k) \geq \hat{b}_{i,j} + u_j(k), & i \in [n], j \in [m], k \in [N] \\
& y_i(k) \geq \hat{c}_{i,j} + \hat{x}_j(k), & i \in [p], j \in [n], k \in [N]
\end{array}
\tag{3.12}
$$

It should be emphasized that this model will most of the times not give a solution $\hat{A}$, $\hat{B}$, and $\hat{C}$ that exactly map the input sequences to the output sequences. Whereas, the MIP model finds the matrices that exactly map the input sequence to the output sequence, it can be the case that the LP gives matrices in which there is some difference between the predicted output sequence $\hat{y}(k)$ and the actual output sequence

$y(k)$. However, in certain cases this LP model can be used to find an MPL system that maps the input sequences correctly to output sequences.

The idea of this LP is that the objective function forces the entries of the matrices $\hat{A}$, $\hat{B}$, and $\hat{C}$ to be as high as possible. Given the fact that we know all $u_j(k)$ and $y_i(k)$, all of the decision variables $\hat{A}$, $\hat{B}$, $\hat{C}$, and $\hat{x}_i(k)$ are bounded from below and above. We will test this model in the next chapter and see what the error is between the predicted output $\hat{y}(k)$ and the actual output $y(k)$.

Chapter 4

Numerical Experiments

This chapter is focused on the numerical experiments that have been conducted regarding the state space identification of MPL systems. We will first introduce the experimental setup and subsequently show the results of the experiments. We will perform tests for the MIQP, MIP, and LP formulation as shown in Chapter 3.

4.1 Experimental Setup

In this section, the experimental setup will be described. We will first introduce how we generate the input-output sequences. Subsequently, we will discuss how the models will be generated, and which experiments we have conducted.

4.1.1 Generating Input-Output Sequences

Before we can solve an LP, MIP or MIQP, we first need to generate correct data. In order to create input-output sequences that correspond to MPL systems, we first need to create the max-plus matrices A, B, and C. During the experiments, we have created matrices of different dimensions. We assumed that the number of inputs and outputs are equal to 1, so we have that $A \in \mathbb{R}_{\varepsilon}^{n \times n}$, $B \in \mathbb{R}_{\varepsilon}^{n \times 1}$ and $C \in \mathbb{R}_{\varepsilon}^{1 \times n}$. In addition, we assumed that there will always be a path from the input $u(k)$ to any resource in the system. This corresponds to the fact that matrix B does not contain any ε entries. For matrix A, we assumed that between 40% and 70% of the entries are equal to ε, and for matrix C this is between 60% and 90%. Here we include the additional constraint that at least one of the entries in all matrices should not be equal to ε. As an example, If the matrix $C \in \mathbb{R}_{\varepsilon}^{1 \times 2}$ and 90% of the values in C should be equal to ε, we set all values equal to ε according to the percentage. However, because of the constraint that at least one of the entries should not equal ε, we set one of the entries equal to a random number. This is to ensure that an output

sequence will be generated. Note that the number of entries equal to ε are most of the times higher for matrix C compared to matrix A. Matrix C maps the relation between $x(k)$ and $y(k)$, which basically means that it only maps the resources that are directly connected to the output $y(k)$. For most systems, only a relatively small percentage of the resources are directly connected to the output. For example, for a system with n-serial resources, this will only be one resource. Matrix A maps the relation between $x(k)$ and $x(k-1)$. From (3.1), we have seen that in a system with n-serial resources, the matrix A has a lower triangular shape. Since a triangular matrix has $\frac{n(n+1)}{2}$ entries, it will have $\frac{100n(n+1)}{2n^2}$ percent of the entries not equal to ε. This corresponds to more than 50%. For max-plus systems with parallel resources, the number of entries will be less. Therefore, we have chosen the number of ε-entries to be between 40% and 70% for matrix A.

If the values in the matrices are not equal to ε, we have assigned a random integer value between 0 and 10. We have chosen a lower bound of 0 in our experiments, because we wanted the values to be nonnegative. This aligns with the underlying physical properties of a system, where it is impossible for a resource to process an item in a negative time duration. The upper bound could in principle be set to any positive value.

After we generated the max-plus matrices, we generated the input sequence $u(k)$ for $k \in \{1, ..., N\}$. In order to generate $u(k)$, we have used the equation $u(k) = u(k-1) + rv(0, 20)$, where $rv(0, 20)$ represents a random integer value ranging from 0 to 20. Her we set $u(0) = 0$. Introducing a random input signal allows us to discover multiple modes of the MPL system, where one time $u(k)$ will be the bottleneck and another time $x(k-1)$ will be the bottleneck. We have set the upper bound of the random value equal to two times the upper bound of the values in the matrices. This is to ensure that $u(k)$ can still possibly be the bottleneck when k has a relatively high value. If the upper bound of the random value would be set equal to the upper bound of the values in the matrices, and one of the values in the matrix A is equal to 10, the resources connected to the high values in the matrix A will almost always be the bottleneck.

Subsequently, the output sequence $y(k)$ for $k \in \{1, ..., N\}$ can be generated by utilizing the matrices A, B, and C, and the input signal $u(k)$ by simply using the basic structure of the MPL system of the form as in (2.1). In the experiments we assumed that all initial buffers are empty, which corresponds to $x_i(0) = \varepsilon$ for all $i \in \{1, ..., n\}$.

4.1.2 Generating the Models

In order to generate all the MIP, MIQP, and LP models, we have used the Gurobi software [17] in a Python environment. For each model, we have exactly implemented the constraints as described in the models (3.9), (3.10), and (3.12). It has to be noted that the MIQP should be able to find an optimal solution where the objective value is equal to 0. This is due to the assumption that there is no noise involved in the processing times or measurements of the data.

4.1.3 The Experiments

All three models have been tested for different dimensions n of the matrices and for a different number of items passing through the system N. In addition, for most of the (n, N)-combinations we have tested it 10 times for each model. Whenever the computation time was below 1 hour on average, we tested it for 10 times. If it was more than 1 hour on average, we have only tested it 3 times. We added a maximum computation time of 3 hours. If the computation time exceeded 3 hours, we did not include those times in the charts of the experimental results. For some of the models we were not able to test for a higher value of n or N, because of the fact that it was too computationally expensive. If we were able to test a given (n, N)-combination for multiple models, we have used the exact same matrices A, B, and C and the exact same input-output sequences $u(k)$ and $y(k)$ for the different models. In Table 4.1, the experiments that have been conducted can be seen. It has to be noted that some of the experiments are overlapping. The experiments of the first three rows in Table 4.1 have been created to compare the three different models for the different values of N. The maximum dimension n that we have tested for the LP, MIP, MIQP are 500, 20, and 10, respectively. For higher values of n, the computation time increased significantly. The experiments in the fourth until the sixth row of Table 4.1 have been created to exactly show the effect of the total number of items N passing through the system for each model individually. We have chosen a dimension n for each model for which we were still able to obtain a solution for $N = 100$ observed by the experiments in the first three rows of the table.

4.2 Experimental Results

In this section, we discuss the results of the experiments presented in Section 4.1.3. It is important to highlight that due to high computing times, not all experiments could be executed.

Model	n	N
LP	[1,...,20,50,100,200,500]	[5,10,20,100]
MIP	[1,...,20]	[5,10,20,100]
MIQP	[1,...,10]	[5,10,20,100]
LP	50	[2,5,10,20,50,100,200,500,1000]
MIP	10	[2,5,10,20,50,100,200,500,1000]
MIQP	5	[2,5,10,20,50,100,200,500,1000]

Table 4.1: Experiments conducted during this research for different models and (n, N)-combinations. n is the number of states in the model (or the dimension of the model), and N is the total number of input-output sequences.

In Figure 4.1, four different line charts can be seen. Each line chart represents the computation time of the MIQP, MIP, and LP for different combinations of the number of items passing through the system N and of dimension n. As can be seen, for all four charts, the computation time of the MIQP is significantly high for relatively low dimensions compared to the MIP and LP. The computation time for the MIP increases significantly for higher dimensions, while the computation time for the LP for a dimension of 20 is still relatively small. The maximum dimension n that we were able to test, decreases as the value of N increases. For example, for the MIQP, we were still able to get results for $n = 10$ and $N = 5$, while we were not able to retrieve the computation time for $n = 8$ and $N = 100$. This is due to the fact that it took too long to find a feasible (for the MIP) or optimal solution (for the MIQP). For one experiment specifically, the MIQP with $N = 20$ and $n = 10$, we observed a computation time of more than 24 hours, and it did not find an optimal solution yet. This was the only experiment where we neglected the restriction of 3 hours maximum out of curiosity. For the experiments conducted, the MIP formulation can solve a system double the order of a system with the MIQP formulation in approximately the same time.

Note that for some of the line charts the computation time decreases while the dimension increases. This is due to some outliers in the data. If the branch-and-cut algorithm branches on the incorrect node and explores all possibilities in that specific node, it can take a very long time before it reaches a feasible or optimal solution. If this will be the case for a lower dimension, it could happen that the computation time is higher than for a higher dimension. As an example, for $N = 5$ and $n = 16$, we have seen a computation time of more than a 1000 seconds. However, for most of the charts we see that the computation time clearly increases as the dimension

grows.

In Figure 4.2, we have specifically depicted the computation time of the LP for $N \in \{5, 10, 20, 100\}$ and different values of n. As we have seen in Figure 4.1, the computation time of the LP was negligible for dimensions smaller than 20. However, we can see that the computation time clearly increases as N and n increase. As was mentioned in Section 3.2.3, the solution $\hat{A}$, $\hat{B}$, and $\hat{C}$ of the LP will not always give the exact mapping from the input sequences to the output sequences. There could be an error between the predicted $\hat{y}(k)$ and the real $y(k)$. In Figure 4.3, four line charts can be seen of the mean squared error of the LP for the different (n, N)-combinations. As can be seen, in some cases the error is quite substantial which indicates that the matrices found do not represent the actual mapping from input to output at all. Also for these charts, all experiments have been tested 10 times, which means that the charts represent the averages over all 10 tests. Even if the average mean squared error over the 10 tests was a higher value than 0, it does not mean that we have not seen cases for which the mean squared error was equal to 0, indicating that the obtained solution exactly mapped the input sequences exactly to the output sequences. In Table 4.2, we have listed the highest dimensions for which we obtained a mean squared error equal to 0 for a given N. It is very interesting to see that we were able to find solutions for relatively high dimensions. In particular, the fact that we found a solution for $N = 100$ and $n = 100$ is very interesing. With the MIP and MIQP it would be almost impossible to find a solution for this N and n due to the computation time required to solve this problem.

N	highest dimension n
5	10
10	17
20	20
100	100

Table 4.2: Table displaying the highest dimension n for which a solution has been found by the LP with a mean squared error equal to 0 for a given N.

In Figure 4.4, three scatter plots can be seen. These plots show the relation between the computation times for the MIQP, MIP, and LP and the number of input-output sequences N. Some of these datapoints have already been seen in the previous plots. These scatter plots clearly show that the computation time of the models increase as N gets bigger.

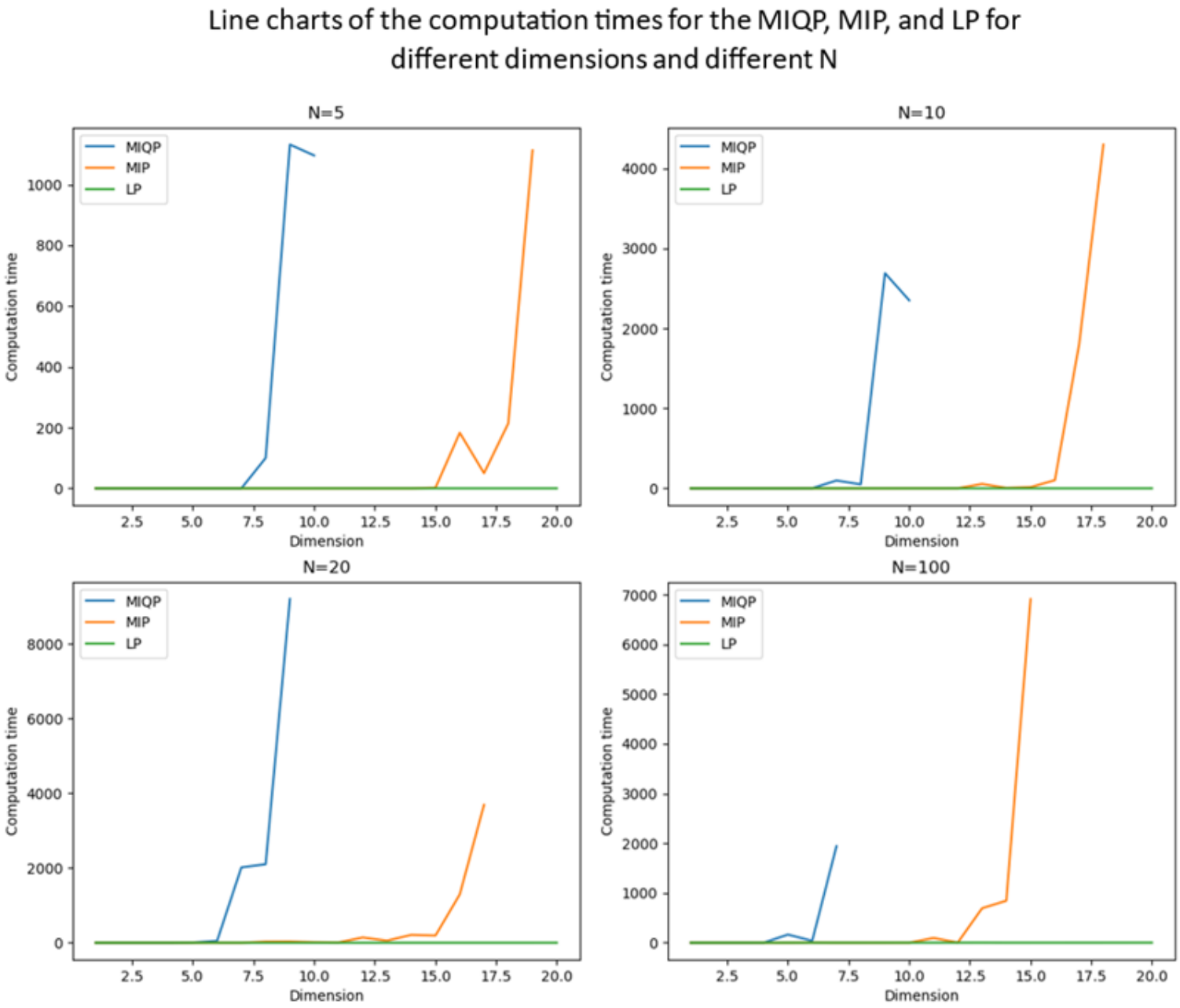


Figure 4.1: Four different line charts for the computation times of the MIQP, MIP, and LP for $N \in \{5, 10, 20, 100\}$. For each chart the computation time is plotted as a function of the dimension of the model. The computation time is in seconds.

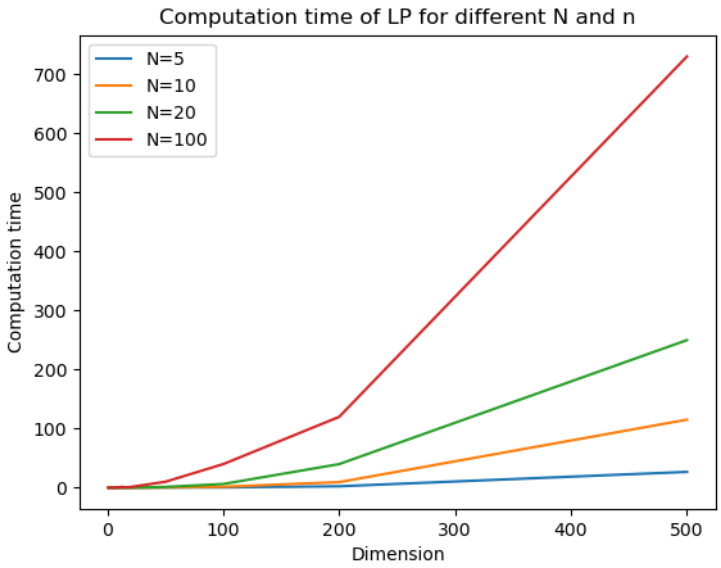


Figure 4.2: Line chart of the computation time for the LP for $N \in \{5, 10, 20, 100\}$. Each line is a function of the dimension of the model. The computation time is in seconds.

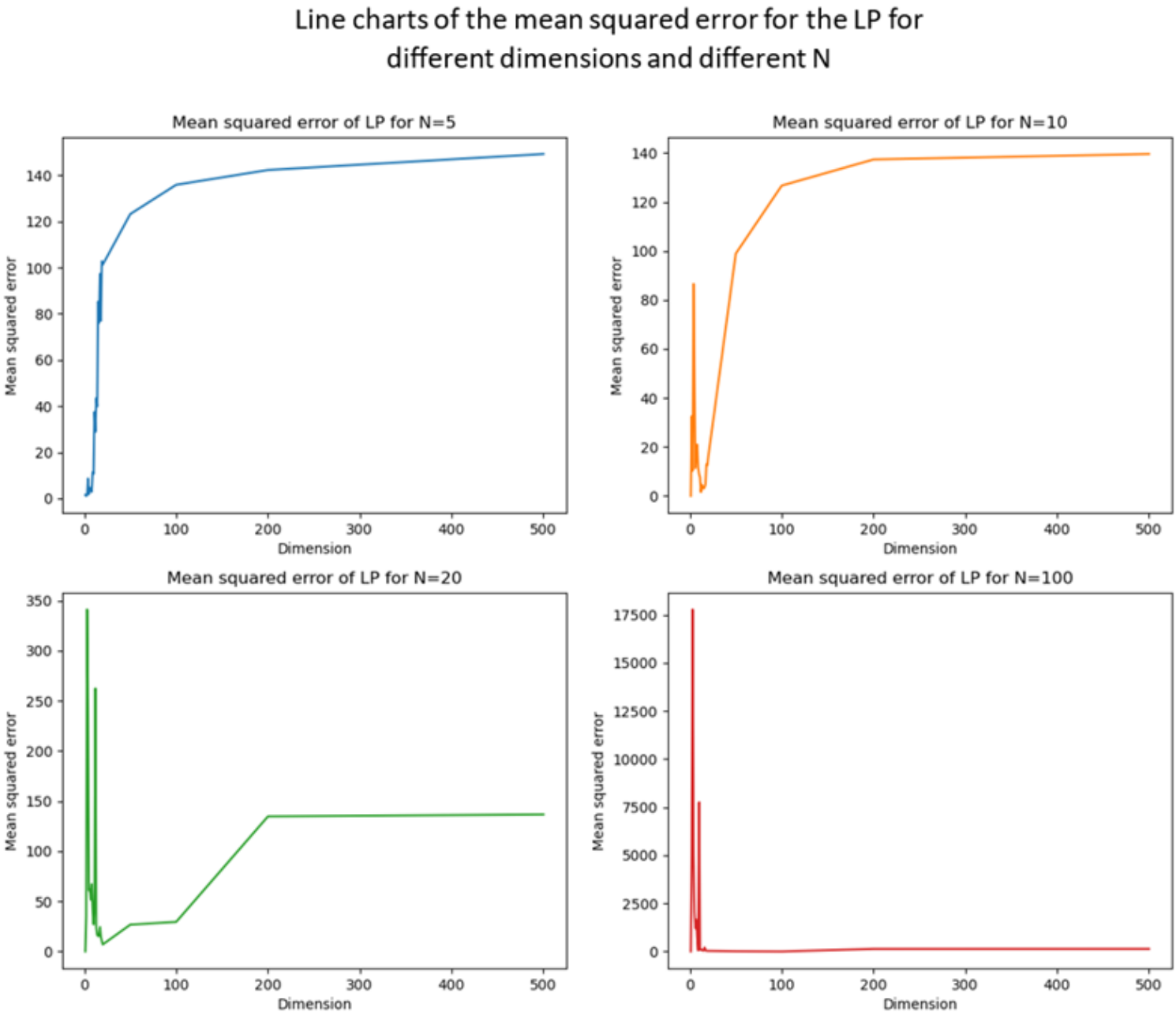


Figure 4.3: Line charts of the mean squared error for the LP for $N \in \{5, 10, 20, 100\}$. Each line chart is a function of the dimension. The computation time is in seconds.

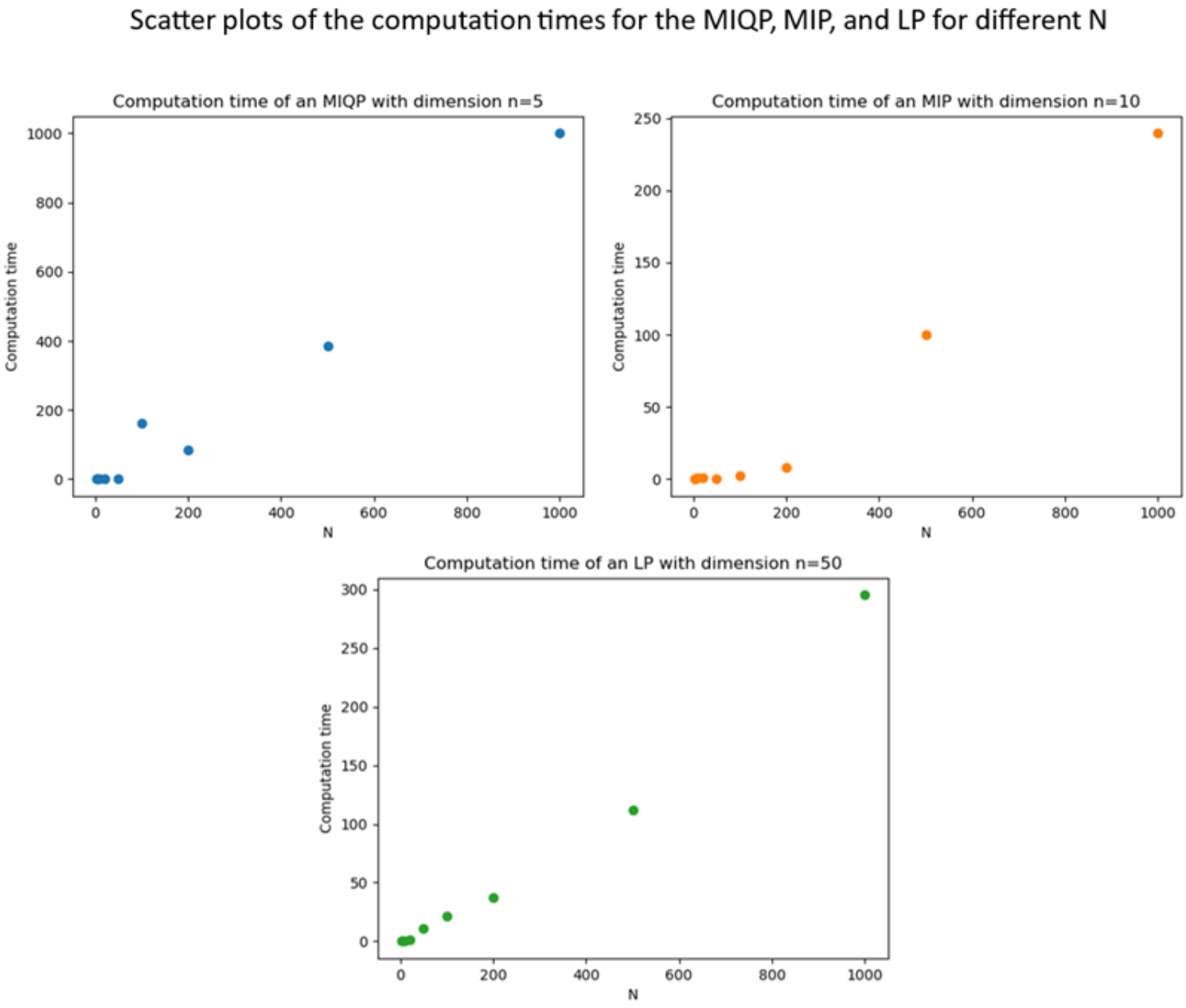


Figure 4.4: Three scatter plots of the computation time of the MIQP, MIP, and LP with a dimension of 5, 10, and 50, respectively. The computation time is in seconds.

Chapter 5

Minimal State Space Realization

This chapter is focused on the minimal state space realization problem in the max-plus algebra, aiming to find the smallest possible MPL system that correctly describes the input-output sequences of a possibly higher-order MPL system. With the order of the MPL system, we are referring to the total number of states in the system. Particularly, the goal is to find a state space realization of the form (2.1) with the smallest possible order n that describes the input-output sequences exactly. In this chapter, we will first provide an example to demonstrate the impact that a minimal state space realization can have on the order of a system. Subsequently, we present an algorithm to find the minimal state space realization of an MPL system.

5.1 Example of Minimal State Space Realization in a Max-Plus Linear System

Suppose we have a system of four serial resources as in Figure 5.1. If we would perfectly model this system using the max-plus algebra, we would need to create a model of order $n = 4$. The model would look as follows:

$$\begin{aligned} x(k) &= \begin{pmatrix} 8 & \varepsilon & \varepsilon & \varepsilon \\ 16 & 2 & \varepsilon & \varepsilon \\ 18 & 4 & 3 & \varepsilon \\ 21 & 7 & 6 & 2 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} 0 \\ 8 \\ 10 \\ 13 \end{pmatrix} \otimes u(k) \\ y(k) &= \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 2 \end{pmatrix} \otimes x(k) \end{aligned} \tag{5.1}$$

However, due to the fact that resource 1 is the bottleneck with a processing time of $T_1 = 8$ and the fact that the processing times of resources 2, 3, and 4 together will never be the bottleneck ($T_2 + T_3 + T_4 = 7$), we can find a minimal state space realization of the system with order $n = 1$. The following MPL system, will result in

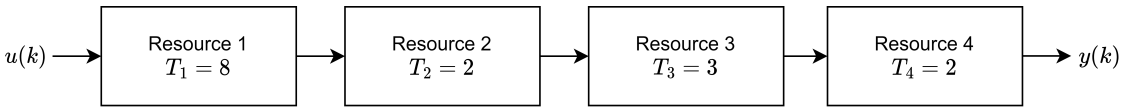


Figure 5.1: System with four serial resources.

the exact same input-output sequence:

$$\begin{aligned} x(k) &= 8 \otimes x(k-1) \oplus 0 \otimes u(k) \\ y(k) &= 15 \otimes x(k) \end{aligned} \tag{5.2}$$

Note that the variable $x(k) \in \mathbb{R}_{\varepsilon}^{1\times 1}$, whereas in the MPL system (5.1), $x(k) \in \mathbb{R}_{\varepsilon}^{4\times 1}$.

As an example, suppose we have an input sequence

$$\{u(k)\}_{k=1}^{N} = 0, 1, 3, 26, 33, 40, 50, 60, 65, 66, 67, 100$$

Running both of the MPL systems of the system in Figure 5.1, will result in the output sequence

$$\{y(k)\}_{k=1}^{N} = 15, 23, 31, 41, 49, 57, 65, 75, 83, 91, 99, 115$$

which could easily be verified. Note again that we assumed that all initial buffers are empty, corresponding to $x_i(0) = \varepsilon$ for all i.

5.2 Minimal State Space Realization Algorithm

The example in the previous section shows that minimal state space realization of a system can have a significant impact on the size of the model. For very simple systems, these realizations could be obtained by hand. However, when the systems become more complex, it could be very time consuming to find a minimal state space realization by hand. In addition, finding a system by hand is very prone to errors. In this section, we will introduce an algorithm to obtain a minimal state space realization given some input-output sequences of an MPL system. The MPL system obtained by this algorithm should exactly map $u(k)$ to $y(k)$ for all $k \in \{1, ..., N\}$. So, the estimates of the output values $\hat{y}(k)$ should exactly equal the observed output values $y(k)$.

If we assume that the measurements or processing times are affected by noise, it is very likely that only an MPL system with a very high-order can describe the input-output sequences exactly, while this high-order model will most likely not represent the underlying system and will be prone to overfitting to the observed input-output sequences. Furthermore, as we have seen in Section 4.2, the computational time

required for the MIQP is already substantial even for relatively low orders. As a result, in the presence of noise, it is not very likely that a feasible solution will be found with an error of zero. If there will be room for error between the prediction of $\hat{y}(k)$ and the observed value $y(k)$, the MIQP (3.10) can be used to find the smallest possible mean squared error between $\hat{y}(k)$ and $y(k)$ for some dimension n. This method will generate matrices A, B, and C for any n, where the mean squared error decreases as n gets bigger and will converge to 0. An acceptable mean squared error will be very specific for each individual case.

However, as in the previous chapters, we make the assumption that processing times and measurements of the input-output sequences are not corrupted by noise. As we have seen in Section 3.2.2, to find the matrices $\hat{A}$, $\hat{B}$, and $\hat{C}$ that correctly describe some input-output sequence, we could use the MIP formulation of (3.9) for some given n. To find the minimal state space realization, we could check what is the smallest n for which the MIP could still find a feasible solution. The method is described in Algorithm 1 below.

Algorithm 1 Minimal State Space Realization via mixed-integer programming

Given: $u(k), y(k) \quad \forall k \in \{1, ..., N\}$
for $n = 1$ to N **do**
  Generate MIP (3.9) with dimension n
  if model found a feasible solution **then**
    $\hat{A}$, $\hat{B}$, $\hat{C}$ found
    Break
  end if
end for

As can be seen, we have tested up until a maximum dimension of N. The dimension of the matrices $\hat{A}$, $\hat{B}$, and $\hat{C}$ that describe the input-output sequence of size N will never have to exceed this value, since this sequence can certainly be described by a max-plus model of order N. In most cases, a model will already be found for a dimension smaller than N. If a model is found, the algorithm stops and returns the model that has been found.

It has to be mentioned that looping over all dimensions, starting at $n = 1$ up until the dimension of the minimal state space realization, and solving an MIP for each of those dimensions, is possibly not the most efficient way to solve this problem. We could think of methods to test specific dimensions and, based on this, narrow down the dimension search space. As an example, we could start of with testing for dimension $n = 3$ and $n = 7$. If no solution is found for the former, but a solution is

found for the latter, we know that the order of the minimal state space realization is between 4 and 7. We could then try to only test dimension $n = 5$ and based on the outcomes of this MIP, continue our search for the minimal state space realization. However, as we have seen in Section 4.2, and given the fact that an MIP problem is NP-hard, the computation time increases significantly when the dimension of the model increases. Consequently, it will be very computationally expensive to test dimensions that are higher than the dimension of the minimal state space realization. However, for these higher dimensions, multiple feasible solutions to the problem exist, which could lead to a faster computation time.

On the other hand, it could also be very computationally expensive to prove that a feasible solution does not exist for a certain dimension. To solve an MIP, Gurobi uses a branch-and-cut algorithm. As a result, to prove that a feasible solution does not exist, it has to continue its search process until all nodes in the branch-and-bound tree are explored.

In order to test the logic explained above, we have performed one numerical experiments that we have tested four times to see if there is any indication that it could be valuable to use some other dimension search space methods. For this experiment, we have generated an input-output sequence of $N = 20$ from four randomly generated eight-dimensional MPL systems. We have generated the data in the same way as explained in Section 4.1.1. We have tested how long it takes the MIP to find a feasible solution or state that no feasible solution exists for MIP (3.9) with dimensions ranging from 1 until 14. The results can be seen in Figure 5.2. For all four tests, we see that if the dimension increases, the computation time increases as well. Especially in test 2, we see that the computation time is very high for dimension $n = 12$ and $n = 14$. In addition, we see that for these tests it did not take a significant amount of time to prove that the model was infeasible. However, as we see, this has only been tested whenever the model had a dimension of $n = 3$ or smaller.

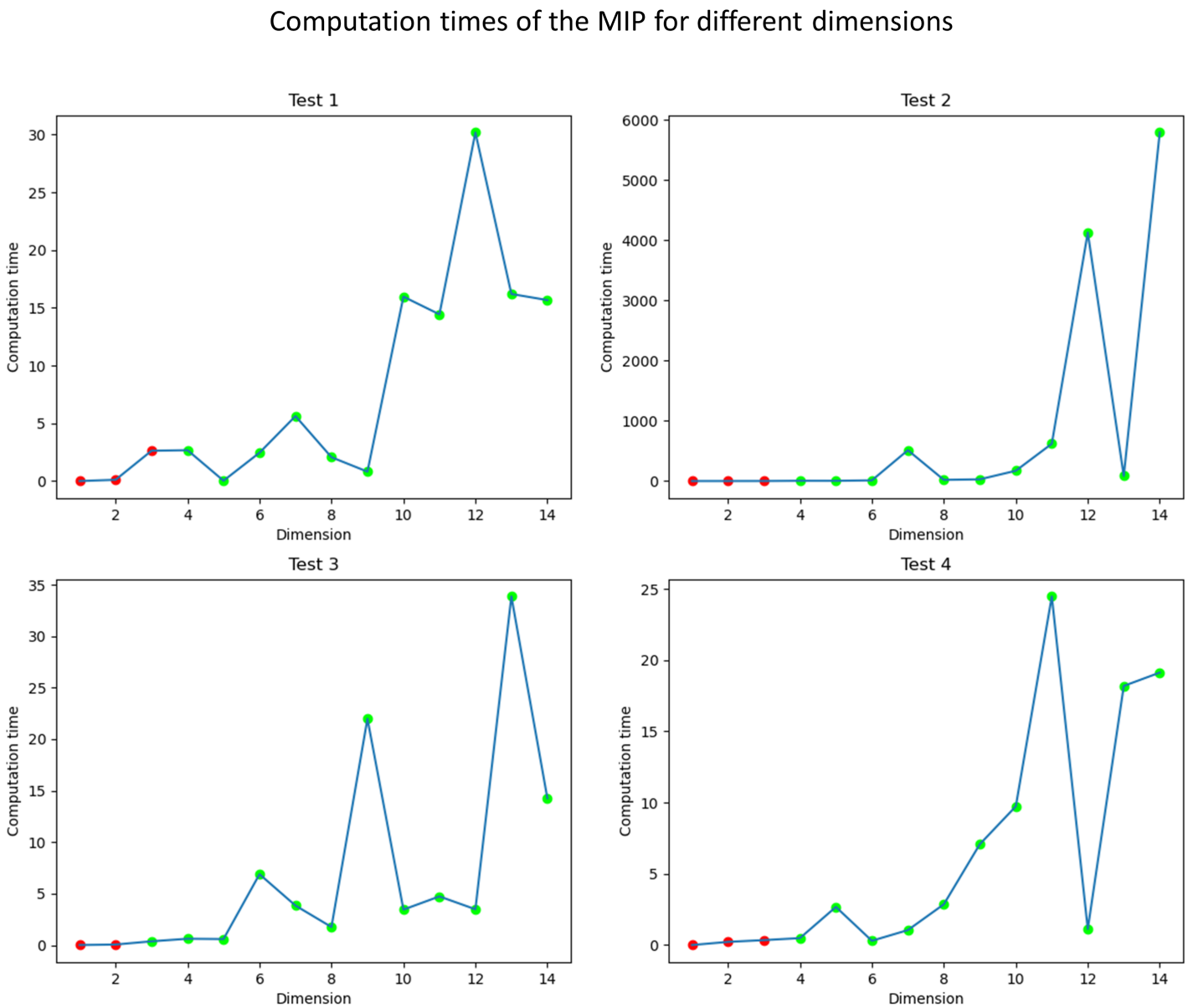


Figure 5.2: Four line charts showing the computation times of the MIP for dimensions ranging from 1 until 14 for four different input-output sequences (for $N = 20$) generated by different randomly generated eight-dimensional MPL systems. A red dot indicates that no feasible solution has been found, while a green dot indicates that a feasible solution has been found. The computation times are in seconds.

Chapter 6

Conclusions & Recommendations

6.1 Conclusions

The developed algorithm for state space identification of MPL systems from input-output sequences using a mixed-integer programming formulation shows a significant improvement over existing algorithms that use a mixed-integer quadratic programming formulation. This holds true, given the assumption that no noise is involved in the processing times or measurements of the data. As was seen in Chapter 4, the dimension of the MPL system could be doubled for the same computation time for the MIP compared to the MIQP. Furthermore, we have seen that the LP formulation was able to find solutions where the inputs were exactly mapped to the outputs even for high dimensions n and for a relatively high number of items passing through the system N. For these high values of n and N, it would be impossible to find a solution with the MIP or MIQP due to the complexity of the problem and the computation time involved with this complexity.

Moreover, we have seen that the MIP formulation can be used to find a minimal state space realization for an MPL system given some input-output sequences. The tests performed in Chapter 5 did not indicate that it will be very useful to optimize the dimension search space. However, more experiments should be conducted with larger dimensions to investigate if it could potentially be interesting.

In addition, in Section 3.1 a method has been developed to model capacity constraints in an MPL system.

6.2 Recommendations

In order to generalize the algorithm for state space identification found in this thesis, it will be interesting to investigate methods in which noise could still be incorporated in the MIP formulation. Possibly, some slack variables could be added to the constraints to compensate the noise. The objective function could then be the sum of all slack variables, where the goal is to minimize this function. This will result in a model with some extra decision decision variables, and this turns the problem into an optimization problem instead of the problem of finding a feasible solution. This will slow down the MIP. However, it could still be modelled as an MIP instead of an MIQP.

In addition, it will be interesting to investigate under which assumptions the LP model will find a solution that exactly maps the input sequences to the output sequences. If the assumptions hold for certain systems, we would be able to find a state space representation of some high-dimensional MPL system for which we would not be able to solve it using the MIP or MIQP formulation.

Furthermore, to improve the minimal state space realization algorithm, methods could be developed to optimize the dimension search space. However, precedingly, tests should be conducted to investigate if it will be valuable to create algorithms for the dimension search space. In these tests, the computation time that it takes to prove that a feasible solution does not exist for a relatively low order should be compared to the computation time of a higher order for the same input-output sequences. In this thesis, we have only seen examples of this for input-output sequences that could be explained by MPL systems of order 3. It will be interesting to test this for input-output sequences where the order is higher. If it will be valuable to find methods to optimize the dimension search space, this method should take into account the trade-off between the computation times of proving that a feasible solution does not exist (if the order is lower than the minimal realization order), and finding a feasible solution (if the order is higher than or equal to the minimal realization order).

Additionally, it could be interesting to investigate whether the LP could be used for the minimal state space realization problem. It would be difficult to state that something certainly is a minimal realization. However, it could possibly be used to find an upper bound on the dimension of the minimal state space realization. Nonetheless, this will only be the case if a solution will be found that exactly maps the input sequence to the output sequence. Before this will be a valuable method, we first need to know under which assumptions the LP will be able to find a solution that correctly maps the input-output sequences.

Bibliography

[1] François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. Synchronization and linearity: an algebra for discrete event systems. 1992.

[2] Jean-Louis Boimond, Laurent Hardouin, and P Chiron. A modeling method of siso discrete-event systems in max algebra. In *European Control Conference ECC'95*, pages 2023–2026, 1995.

[3] B Ciffler. Scheduling general production systems using schedule algebra. *Naval Research Logistics Quarterly*, 10(1):237–255, 1963.

[4] Guy Cohen, Didier Dubois, J Quadrat, and Michel Viot. A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE transactions on Automatic Control*, 30(3):210–220, 1985.

[5] RA CUNINGHAME-GREEN. Minimax algebra. *Economics and Mathematical Systems*, 1979.

[6] Raymond A Cuninghame-Green. Describing industrial processes with interference and approximating their steady-state behaviour. *Journal of the Operational Research Society*, 13(1):95–100, 1962.

[7] Bart De Schutter. Minimal state-space realization in linear system theory: an overview. *Journal of computational and applied mathematics*, 121(1-2):331–354, 2000.

[8] Bart De Schutter and Bart De Moor. The extended linear complementarity problem. *Mathematical Programming*, 71(3):289–325, 1995.

[9] Bart De Schutter and Bart De Moor. Minimal realization in the max algebra is an extended linear complementarity problem. *Systems & Control Letters*, 25(2):103–111, 1995.

[10] Bart De Schutter and Bart De Moor. Matrix factorization and minimal state space realization in the max-plus algebra. In *Proceedings of the 1997 American*

control conference (Cat. No. 97CH36041), volume 5, pages 3136–3140. IEEE, 1997.

[11] Bart De Schutter and Ton van den Boom. Max-plus algebra and max-plus linear discrete event systems: An introduction. In *2008 9th International Workshop on Discrete Event Systems*, pages 36–42. IEEE, 2008.

[12] Bart De Schutter, Ton JJ van den Boom, and Vincent Verdult. State space identification of max-plus-linear discrete event systems from input-output data. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 4024–4029. IEEE, 2002.

[13] Samira S Farahani, Ton van den Boom, and Bart De Schutter. An approximation approach for identification of stochastic max-plus linear systems. *IFAC Proceedings Volumes*, 44(1):8241–8246, 2011.

[14] Samira S Farahani, Ton van den Boom, and Bart De Schutter. Exact and approximate approaches to the identification of stochastic max-plus-linear systems. *Discrete Event Dynamic Systems*, 24:447–471, 2014.

[15] F Gallot, Jean-Louis Boimond, and Laurent Hardouin. Identification of simple elements in max-algebra: application to siso discrete event systems modelisation. In *1997 European Control Conference (ECC)*, pages 1866–1871. IEEE, 1997.

[16] Elmer G Gilbert. Controllability and observability in multivariable control systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2):128–151, 1963.

[17] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[18] Laurent Hardouin, Bertrand Cottenceau, Ying Shang, Jörg Raisch, et al. Control and state estimation for max-plus linear systems. *Foundations and Trends® in Systems and Control*, 6(1):1–116, 2018.

[19] Bernd Heidergott, Geert Jan Olsder, Jacob Van Der Woude, and JW van der Woude. *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*, volume 13. Princeton University Press, 2006.

[20] Magnus R Hestenes, Eduard Stiefel, et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

[21] BL HO and Rudolf E Kálmán. Effective construction of linear state-variable models from input/output functions. *at-Automatisierungstechnik*, 14(1-12):545–548, 1966.

[22] James Hook. Max-plus linear inverse problems: 2-norm regression and system identification of max-plus linear dynamical systems with gaussian noise. *Linear Algebra and its Applications*, 579:1–31, 2019.

[23] Aiwen Lai, Sébastien Lahaye, and Alessandro Giua. State estimation of max-plus automata with unobservable events. *Automatica*, 105:36–42, 2019.

[24] Lennart Ljung. *System identification*. Springer, 1998.

[25] Eric Menguy, Jean-Louis Boimond, Laurent Hardouin, and Jean-Louis Ferrier. A first step towards adaptive control for linear systems in max algebra. *Discrete Event Dynamic Systems*, 10:347–367, 2000.

[26] Bruce Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE transactions on automatic control*, 26(1):17–32, 1981.

[27] Gernot Schullerus and Volker Krebs. Diagnosis of batch processes based on parameter estimation of discrete event models. In *2001 European Control Conference (ECC)*, pages 1612–1617. IEEE, 2001.

[28] Gernot Schullerus and Volker Krebs. A method for estimating the holding times in timed event graphs. In *Sixth International Workshop on Discrete Event Systems, 2002. Proceedings.*, pages 209–216. IEEE, 2002.

[29] Gernot Schullerus, Volker Krebs, Bart De Schutter, and Ton van den Boom. Input signal design for identification of max-plus-linear systems. *Automatica*, 42(6):937–943, 2006.

[30] A Seleim and H ElMaraghy. Generating max-plus equations for efficient analysis of manufacturing flow lines. *Journal of Manufacturing Systems*, 37:426–436, 2015.

[31] Leonard Silverman. Realization of linear dynamical systems. *IEEE Transactions on Automatic Control*, 16(6):554–567, 1971.

[32] Srinivas Sridharan. Deterministic filtering and max-plus methods for robust state estimation in multi-sensor settings. *arXiv preprint arXiv:1211.1449*, 2012.

[33] Ton van den Boom and Bart De Schutter. A stabilizing model predictive controller for uncertain max-plus-linear systems and uncertain switching max-plus-linear systems. *IFAC Proceedings Volumes*, 44(1):8663–8668, 2011.

[34] Ton JJ van den Boom, Bart De Schutter, and Vincent Verdult. Identification of stochastic max-plus-linear systems. In *2003 European Control Conference (ECC)*, pages 618–623. IEEE, 2003.