Modeling and Control of Switching Max-Plus-Linear Systems

Rescheduling of railway traffic and changing gaits in legged locomotion

B. Kersbergen

Modeling and Control of Switching Max-Plus-Linear Systems

Rescheduling of railway traffic and changing gaits in legged locomotion

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben;
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
22 oktober 2015 om 10:00 uur

door

Bart Kersbergen
Ingenieur in systeem- en regeltechniek,
Technische Universiteit Delft
geboren te Nieuwegein, Nederland

This dissertation has been approved by the
promotor: Prof.dr.ir. B. De Schutter
copromotor: Dr.ir. A.J.J. van den Boom

Composition of the doctoral committee:

Rector Magnificus	chairman
Prof.dr.ir. B. De Schutter	Delft University of Technology
Dr.ir. A.J.J. van den Boom	Delft University of Technology

Independent members:

Prof.dr.ir. R.P.B.J. Dollevoet	Delft University of Technology
Prof.dr. B.F. Heidergott	Vrije Universiteit Amsterdam
Dr. A. D'Ariano	Università degli Studi Roma Tre
Dr. R.M.P. Goverde	Delft University of Technology
Prof.dr.ir. J. Hellendoorn	Delft University of Technology, reserve member

Other member:

Dr. G.A.D. Lopes	Delft University of Technology

This thesis has been completed in partial fulfillment of the requirements of the Dutch Institute of Systems and Control (DISC). This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs.

TRAIL Thesis Series T2015/16, The Netherlands, TRAIL Research School

Published and distributed by: B. Kersbergen
E-mail: bartkersbergen@gmail.com

ISBN: 978-90-5584-196-7

Keywords: Railway traffic management, discrete-event systems, max-plus-linear algebra, switching max-plus-linear systems, distributed model predictive control, mixed integer linear programming, legged locomotion, gait switching.

Printed in the Netherlands.

Preface

The research presented in this thesis has been conducted as part of the STW project "Model-Predictive Railway Traffic Management". The project consists of work done at the Department of Transport & Planning by Pavle Kecman and his supervisors dr. Rob Goverde and prof.dr.ing. Ingo Hansen, and at the Delft Center for Systems and Control of the Delft University of Technology by the author of this thesis and his supervisors dr.ir. Ton van den Boom and prof.dr.ir. Bart De Schutter. The goal of the project was the development of new models and a new (predictive) control approach for anticipative management of railway networks. To achieve this goal the research was split up into three subjects: Monitoring methods that actively monitor trains in the network, detect and predict conflicts, and determine up-to-date process times and estimate future process times, models of the railway traffic that can be updated continuously with the latest information and selected control actions, and model predictive controllers that optimizes future control decisions by using predictions of the future behavior of the railway traffic. The research in this thesis focuses on the model predictive controllers and the model used in the model predictive controller. The research performed at the Department of Transport & Planning was aimed at developing the monitoring methods and models for predictive railway traffic management. The work of Pavle Kecman has already been published in "Models for Predictive Railway Traffic Management" [53].

First and foremost, I would like to thank my daily supervisor Ton van den Boom. Whenever we discussed my research or just made small talk he was always very helpful and enthusiastic, thinking up new ideas, or telling about one of his many commuting experiences on the train. The support and guidance he gave me really helped me finish this my research. It has been a pleasure working with him. I would also like to thank my promotor Bart De Schutter. He always made sure I kept the end goal in mind and made sure I would not get lost or stuck (for too long) on the way there. He always made sure he had time to read my unpublished work and provide plenty of feedback. The feedback sometimes felt a little overwhelming but, after processing it, always resulted in much better papers.

I would like to thank the other members of the project, namely Pavle Kecman, Rob Goverde, and Ingo Hansen. Pavle, thank you for having time to discuss my research and ideas from time to time, help prepare for the STW users' group meeting, and having fun during the RailCopenhagen conference. Rob and Ingo, thank you for sharing your vast knowledge and experience on everything related to railways and for showing a different perspective on my work from time to time. Furthermore I would like to thank the members of the users' committee of the project: Bob Janssen, Leo Kroon, Edo Nugteren, Alfons

Schaafsma, Ello Weits, and Jianxin Yuan for their continued dedication to the project and their support and feedback.

In the last four years I have shared the office with several others: Yashar, Ismini, Dang, Amol, Alfredo, Max, Hildo, Jun, and Armand. Thank you all for putting up with me, sharing your knowledge, helping me out, and for making the office a friendly and enjoyable environment. I would also like to thank our secretariat: Kitty, Heleen, Marieke, and Kiran for all their help and support.

There have been occasions where we have had time to engage in social activities outside of work and I thank Sachin, Hans, Paolo, Laurens, Hildo, Edwin, Renshi, Subramanya, Elisabeth, Reinier, and Baptiste for inviting me to the game nights and having fun with me.

The first conference you go to can be quite an event. I was glad I was not the only one going there and I am grateful to Yihui Wang for letting me tag along during the conference.

Another conference was in Qingdao, China, and I would like to thank Zhe Cong for traveling with me to Qingdao and helping me find my hotel. Furthermore I would like to thank Yu Hu for showing some of the most interesting spots of Qingdao. I would also like to thank Le Li for letting me tag along to the dinner with your old classmates in Qingdao, and for showing me the city of Beijing and going to the Great Wall of China with me. I really enjoyed my time in China and greatly appreciate that you were all willing to show me a small part of China.

I would also like to thank Max Potters for the early morning talks. A friendly chitchat in the office at 7:00 in the morning is a good way to start working. I would also like to thank Mohammad Hajiahmadi and Yu Hu for their time in answering my questions about the LaTeX template for this thesis and other questions relating to the doctoral regulations.

I want to thank my friends for all the great times we have had these last years, from summer holidays to Italy and Greece, to ski trips to Winterberg, for the times we went out for drinks together and had a good time, for the fun we had playing volleyball, and for all the other times.

Finally I would like to thank my parents, Jan and Lineke, and my brother and sister, Frank and Janneke, for all the support and love they have given me all these years and for always believing in me.

Bart Kersbergen,
Delft, October 2015.

Contents

Chapter 1

Introduction

Often stabilizing or improving the dynamical behavior of a system is the goal when designing a controller. But for some systems the timing between discrete events is more important than the dynamical behavior between the events. Such systems can be described as discrete event systems. Especially in manufacturing, transport, and logistics such systems are often found.

Examples of discrete event system are the scheduler of the print tasks in an industrial printer, the job scheduler in a manufacturing plant, a baggage handling system, a passenger railway system, and the gait scheduler for a legged robot. The scheduler of the printer determines the order in which the print tasks are processed in the printer such that the throughput is optimized. In a manufacturing plant with multiple machines that can perform different tasks, the job scheduler makes a schedule and updates it during the day to maximize the productivity of all machines. Baggage handling systems can be found in almost all airports and ensure that the enormous amount of bags are transported from the baggage drop off points at the airport to the planes and from the planes to the baggage retrieval areas in the airports. There are kilometers of tracks in the airport to move all those bags and an automated system determines the routes and the order in which the bags are transported to ensure all bags arrive at their destination in time. The trains of a passenger railway system can be modeled by the arrival and departure times at stations and junctions. These events occur in a specific order and at set times determined by the timetable. For a legged robot the gait, or pattern of movement of the legs, can be described by the touchdowns and lift-offs of the feet from the ground and the order in which these events occur.

A system that can be described by a max-plus-linear model can be characterized as a discrete event system in which only synchronization occurs, but no concurrency or choice [3]. In some of these systems the order of the events can be changed, they can then be described by switching max-plus-linear systems. In this chapter we give an introduction to the two applications of switching max-plus-linear systems that will be the subject of this thesis: railway traffic management and legged locomotion. In Section 1.1 railway traffic management will be discussed shortly, including a short review of the different approaches used for on-line railway traffic management in literature. In Section 1.2 the subject of legged locomotion is introduced as well as the different methods of modeling

legged locomotion in the literature. The outline of this thesis is given in Section 1.3.

1.1 Introduction to railway traffic management

In many countries in the world large, complex, and very busy railway networks have been built. Especially in North and West Europe, China, and Japan the railway networks are used near their maximum capacity. As a result, very little buffer time is available to recover from delays.

Every day small delays occur in almost all railway networks, such small delays are often called "disturbances" in literature. In literature "disruptions" are large perturbations, such as trains breaking down and tracks being blocked, causing trains to be canceled.

In order to deal with disturbances dispatchers reschedule and reroute trains, or break connections. Currently most dispatchers take these decisions based on their experience, a given set of ground rules, and a limited overview of the network situation.

To be able to handle disruptions trains may need to be canceled, they may need to be rerouted through the entire network. These changes affect the rolling stock circulation and the personnel schedules. As a result the rolling stock circulation needs to be recomputed and adjustments to the personnel schedules need to be made.

In this thesis we focus on railway traffic management for disturbances, and therefore we will not consider the rolling stock circulation or the personnel schedules. For an overview of those research areas, literature on disruption management, and integrated approaches combining several of these research areas the reader is referred to the survey paper of Cacchiani et al. [7].

The literature on methods to railway traffic management for disturbances can, for the most part, be split up into two groups based on the size of the problem instances they consider: there are approaches that focus on a small part of the railway network and there are approaches that take the entire (national) network into account. We will call railway management for small parts of the railway network "local control" and railway management for the entire network "global control".

For both the local and global control models of the trains and railway network are used to predict the effect of the rescheduling actions on future arrival and departure times of the trains. The models used by the local and global controllers can have different levels of detail. There are microscopic models which consider as many details as possible, macroscopic models which only model the most important details, and mesoscopic models which model some parts in great detail and some parts with only the most important details depending on the application of the model. Next, we will discuss the microscopic and macroscopic models.

1.1.1 Microscopic and macroscopic models

Consider the small network in Figure 1.1. It is shown in microscopic detail in the top half and in macroscopic detail in the bottom half. In the microscopic model the tracks are

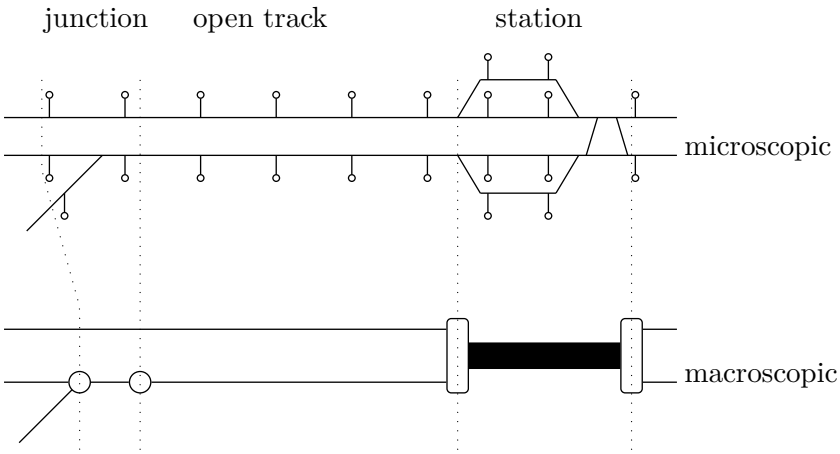

Figure 1.1: Small example railway network with one station, a junction, and a track connecting the two.

built up from block sections and include the signals that are used for the safety system and indicate whether a block section is available. The length, gradient, and maximum velocity of each block section is given. Each signal of the signaling system is modeled and operates according to the signaling rules of the network. Trains have to stop for red signals and yellow signals are used to indicate that the next signal is currently red and the train should start breaking. For each station all platforms are considered separately and all block sections of all tracks are considered. In the interlocking area of stations the signaling system has specific rules to ensure no conflicting movements of trains occur. This is modeled by a limited number of paths the trains can chose and excluding paths that cause conflicts with other trains. In some cases the train dynamics are also modeled using a dynamic model of the train. The model of the train is based on the length, weight, the train resistance, and the power of the locomotive. The acceleration, deceleration, and velocity of the train also depends on the characteristics of the block section.

In many of the decision support systems the train dynamics and block section characteristics are used to determine the running times of all trains for all block sections they traverse off-line. These running times are then used as fixed times during the on-line rescheduling where only the departure times of the trains are changed, but the trains drive according to the off-line determined speed profile and running times. In some of the railway traffic management methods the running times are computed during the rescheduling and for every new schedule and new route the running times are recomputed [22].

In the macroscopic model the block sections of the tracks are modeled as a single section for each track with averaged characteristics. The signaling system is not modeled explicitly, but instead headway times between trains are used. Tracks and platforms in stations are not modeled, only a point for arrival and a point for departure and a link connecting them are modeled. Routes through station areas are not considered, and in some cases the capacity at stations is not considered either, but it is assumed to be sufficient such that each train that arrives can enter the station and stop at a platform. The junction is modeled in the same way as the station.

Microscopic models can be used to determine schedules and routes through station

areas and smaller networks, but for large or nationwide networks these models quickly become too complex for the purpose of railway traffic management. Since macroscopic models are less detailed even models of nationwide networks can be used in railway traffic management. The routes through station areas however cannot be determined by these macroscopic models. If a different route is set, the time for the train to traverse through the station may change, which will affect the results. In this case the process times in the macroscopic model needs to be adjusted.

Some of the first papers where control of the railway systems in order to achieve optimal schedules are de Waal et al. [28], Minciardi et al. [70]. Since then developing methods to determine optimal schedules and routes for trains in the case of disturbances has become a topic many researchers have been working on. This field of research will be discussed next.

1.1.2 Railway traffic management methods

In this subsection we will discuss various railway traffic management methods, for the rescheduling and rerouting of trains, that are currently being developed or have been developed in recent years. We will make a distinction between methods for local control and global control.

Many methods for local control have been proposed in the recent years such as the methods of Caimi et al. [8], Corman [10], Corman et al. [12, 15], D'Ariano [21], D'Ariano and Pranzo [22], D'Ariano et al. [23], Rodriguez [77].

Caimi et al. [8] developed a railway traffic management method that tries to schedule and route all trains in an area in and around a large station. A model predictive control approach with a microscopic model of the railway operations is used based on blocking times. At each point where rescheduling of a train is possible, a set of possible blocking times for different routes and departure times at platforms or arrival times at the boundaries of the area are considered for that train. As many trains as possible are then assigned a route with corresponding blocking times while all safety and operational constraints are respected. The objective is to optimize the passenger satisfaction, measured by punctuality and reliability. The resulting optimization problem is a binary linear programming problem.

Rodriguez [77] proposes a method for the routing and scheduling of trains through an area around the Pierrefitte-Gonesse junction north of Paris. They use a microscopic simulation to model the train and driver behavior and describe the routing and scheduling problem as a constraint programming problem. For the given case study they are able to reduce the delays between 63 and 96%.

In the work of D'Ariano [21], D'Ariano and Pranzo [22], D'Ariano et al. [23] the railway operation is also modeled as a microscopic model based on blocking times with an Alternative Graph approach. In their alternative graph approach for every train occupying a block section a node is created in the graph. The nodes of a single train are then connected to each other through running time constraints and for every pair of trains occupying the same block section headway/separation constraints are added. If the order in which the trains can occupy the block sections can be changed with rescheduling actions,

then a pair of alternative arcs defining the two orders in which the trains can occupy the block section are added to the graph. A new schedule for the railway traffic is found when for each pair of alternative arcs only one arc is chosen and no circuits of positive length are present in the graph. The graph has an extra node, to which all nodes are connected and the weights of the arcs from all nodes to this extra node are chosen such that minimizing the maximum weight of all paths from the starting to the ending node corresponds to minimizing the maximum consecutive delay. To solve this problem the authors use their own branch and bound algorithm.

Corman [10], Corman et al. [12, 15] have extended the work of D'Ariano et al. [23] to also consider breaking connections. For different sets of maintained connections the maximum consecutive delay is determined and the decrease of the maximum consecutive delay is weighted against the number of broken connections. The biggest differences in modeling between this work and our work is the level of detail considered and the solver used to solve the problem. We do not consider block sections, but only tracks between stations and the interlocking area of a station is considered as a single node. Our models are also built as cyclic models allowing us to easily expand the simulation and control period to multiple cycles without having to rebuild the entire model. The method for solving the optimization problem is also different. D'Ariano et al. [23] and Corman et al. [15] use a specifically designed branch and bound algorithm made for minimizing the maximum consecutive delay. In most cases the methods for local control use microscopic models. By using microscopic models they can model the routes, the signaling and safety system, and the speed profiles of the trains. In some cases they can even change the routes and adjust the speed profiles of the trains. Because of the microscopic models the networks they consider must be relatively small for the railway traffic management methods to be able to find solutions quickly.

Several researchers made strides to extend their work to large scale networks, or developed new methods for large scale networks such as Corman [10], Corman et al. [13, 14, 16], Kanai et al. [51], Kecman et al. [56], Törnquist [82], Törnquist and Persson [83], Törnquist-Krasemann [84].

Corman [10], Corman et al. [13, 14, 16] extend their previous work to multiple areas using a supervisory controller that coordinates between the areas. For smaller instances and a limited number of areas the bi-level approach works well, but once the prediction horizon becomes larger and the number of areas increases a good feasible solution is not always found.

Törnquist and Persson [83] proposes a railway traffic management method for large network instances, covering a part of the Swedish railway network in the south of the country. Specifically they consider a network with multiple parallel tracks that can be used between stations. They assume that there are no restrictions on the tracks the trains can use. They propose a mixed integer programming problem to schedule the trains and divide them over the tracks such that the delays are minimized. For some instances the MIP solver cannot find a feasible solution within the available time. To solve this problem they propose a heuristic method to solve the problems in [82]. Törnquist-Krasemann [84] extends the model and solution procedures to also consider the available routes through

the stations limiting the number of available tracks for each train.

Kecman et al. [56] uses the same Alternative Graph approach as Corman et al. [13, 14, 15, 16], D'Ariano and Pranzo [22], D'Ariano et al. [23] but now for macroscopic models with different levels of details. The level of detail in this model is similar to our own work. One of the differences between their work and ours is that they model the system as an alternative graph and not as a max-plus-linear system and the alternative graph is specifically designed such that the maximum consecutive delay can be minimized using the branch and bound algorithm of D'Ariano et al. [23]. This branch and bound algorithm is specifically designed to minimize the weight of the longest path and the alternative graph is adjusted such that the weight of the longest path corresponds to the maximum consecutive delays. If they want to consider breaking connections they have to adjust the whole solution procedure as was done by Corman et al. [15].

Kanai et al. [51] propose a tabu search method to determine the connections of the trains for the entire network. The goal is to minimize the passenger dis-utility For this they propose various linear and non-linear objective functions. By simulating the railway traffic and the passenger behavior they are able to determine the passenger dis-utility for a given set of connections. The tabu search algorithm optimizes the set of connections to minimize the dis-utility They test their tabu search on a case study consisting of a part of the Japanese railway consisting of 41 stations and 40 trains. For this test case the process time of the algorithm was about 6 minutes in all scenarios using a PC with an Intel Core2Duo CPU.

Many more papers have been published on this subject and several review papers on this subject have recently been published: Cacchiani et al. [7], Corman and Meng [11], Fang et al. [30]. The authors of these review papers conclude that much research has been done in the recent years on railway traffic management, but most of it has been on small networks or parts of a network. There is a need for railway traffic management methods for large scale networks and possible directions are the use of macroscopic models and multi-level control.

The research in this thesis on railway traffic management is based on the framework given in Figure 1.2. In the framework it is assumed that several local controllers determine the routes and trajectories of the trains in small parts of the network. The global model predictive controller determines an updated timetable for the entire network by changing the order of the trains on the tracks and by changing the departure and arrival times at stations and junctions such that the delays in the network are reduced. The updated timetable is given to the local controllers and used to set local and boundary conditions that ensure global feasibility. The local controllers can interact with the global controller when the updated timetable is not feasible for their part of the network. Based on this feedback the global controller updates its information and updates the timetable again. The monitoring part of the framework tracks the location of the trains and monitors the situation of the network. Furthermore, it predicts future conflicts, process times, and arrival and departure times and provides this information to the model predictive controller.

In this thesis we concentrate on the global model predictive controller and the model

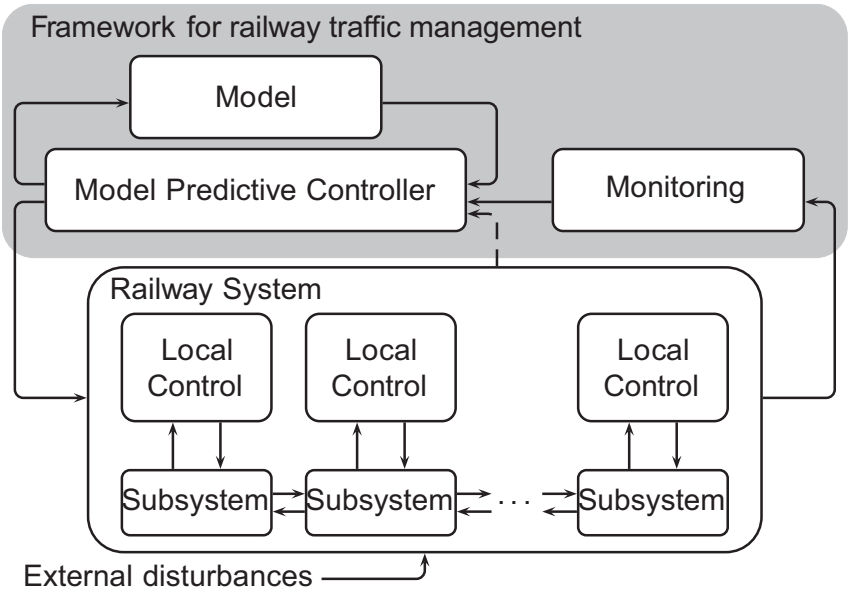

Figure 1.2: Framework for railway traffic management.

used to predict the future arrival and departure times of the trains. The goal is to develop a railway traffic management system for the entire Dutch railway network. We will use a macroscopic model for the prediction of the future arrival and departure time that can be described by a switching max-plus-linear model. To improve the computation time of the model predictive controller a method is proposed to convert and reduce the macroscopic model in size. Furthermore several distributed model predictive control approaches are proposed and compared.

Currently the monitoring of the trains in the Netherlands is done based on data from the signaling system and the train describer system. In the future this may be replaced by systems using a global positioning system for even more accurate information. The research for the monitoring system and the prediction of future process times has been done by Kecman [53], Kecman and Goverde [54, 55], while the local control can be done by one of the local railway traffic management methods described earlier in this section, possibly in combination with trajectory planning [4, 89–91] in the future.

1.2 Introduction to legged locomotion

Legged robots are becoming increasingly prominent in the robotics field. Their advantages on unstructured terrain combined with the challenges in mechatronics and control have fueled a community of academics and industry alike that aims to build truly autonomous legged robots with agility akin to animals. The recent successes by Boston Dynamics on quadrupeds [74], and the efforts of the Japanese community on developing home assistance anthropomorphic robots, such as humanoid robots developed by Honda [46] or ASIMO [78], or robots such as HRP [52], contribute to this growing interest in legged robots.

A fundamental element in the control of a legged robot is the synchronization of its

legs. For bipedal robots synchronization is usually addressed implicitly, since balancing is the biggest challenge [36, 63, 96]. For robots with more than two legs, many different locomotion patterns can be chosen, resulting in the number of distinct gaits increasing with the number of legs (see Holmes et al. [47] for an extensive review on the elements of dynamic legged locomotion). The second part of this thesis focuses on the systematic design of gait controllers for robots with many legs where the number of available gaits is high. From a control design point of view, legged locomotion can be implemented via a *gait reference generator* module and a *dynamic tracking controller* module, as illustrated in Figure 1.3.

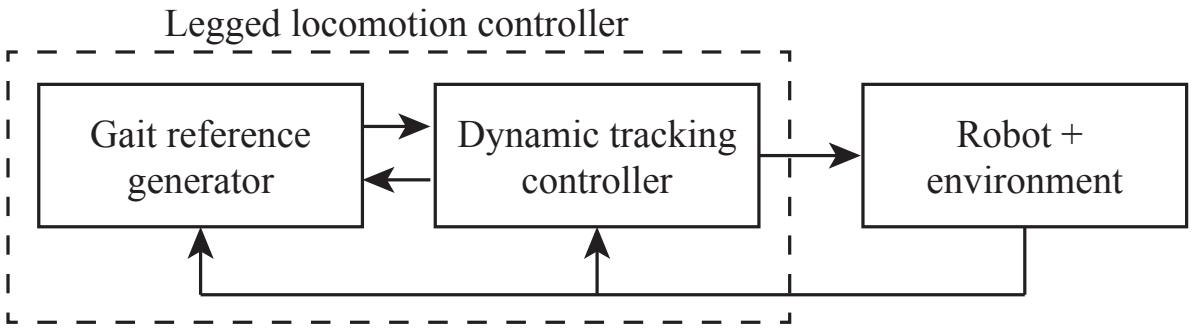

Figure 1.3: The standard partitioning of a legged locomotion controller. The gait reference generator subsystem provides reference signals to the tracking controller. Feedback can exist from both the robot and the tracking controller to the gait reference generator.

The gait reference generator is a component that generates cyclic reference signals in a synchronized way, and the dynamic tracking controller translates the typically low-dimensional reference signals into the high-dimensional motion of the robot's limbs and implements other desirable dynamical properties such as balancing, see e.g. Vukobratovic and Borovac [88]. The advantage of the partition into a gait reference generator and a dynamic tracking controller is that the gait reference generator can be designed without explicit knowledge of the mechanics of the robot (other than the number of legs) while the latter is designed specifically for each robot model.

Most gait reference generator designs are based on Central pattern generators (CPGs) (see Ijspeert [48] for a survey on CPGs). CPGs are neural networks found in animals that can generate complex periodic signal patterns. They are called *central* pattern generators because they do not require sensory feedback to produce the patterns. In animals they generate rhythmic patterns for movement. So CPGs offer a natural bio-inspired control framework that addresses locomotion patterns.

Although widely used, CPGs offer their own set of challenges because of their mathematical formulation as sets of coupled differential equations. One of those challenges is the transient behavior that exists during gait transitions. Gait transitions are a very natural occurrence in nature; animals change gait to accommodate for different types of terrain, locomoting speeds, and to minimize the energy needed to move at the desired speed. As in normal systems modeled by differential equations, the transient behavior is typically less understood than the steady-state behavior. A lot of researchers have

worked on gait transition in the CPG framework (see Aoi et al. [1], Daun-Gruhn and Toth [24], Inagaki et al. [49, 50], Li et al. [62], Nagashino et al. [72], Santos and Matos [79], Zhang et al. [95], and the references within [48]). Other work on gait transition without using CPGs in the continuous-time domain has been performed by Haynes and Rizzi [41], Haynes et al. [43]. The traditional approach for gait transition in the CPG framework exploits the bifurcations that occur when changing parameters in the set of coupled differential equations. This can lead to intricate analysis of the global behavior due to the continuous-time models used. As a result, changing the locomotion pattern in a short time and without the robot falling over is a difficult task that is hard to implement on-line.

An alternative approach to CPGs for the synchronization of cyclic systems is called the "Buehler clock" [80]. Related work has also been published in [64]. In this framework, the control structure is built up as shown in Figure 1.3. The gait reference generator generates piecewise constant phase velocity reference signals. The dynamic tracking controller maps the phases and velocities to the movement of the feet and ensures the feet move according to the reference signals. The advantage of the Buehler clock is that, since it is constructed as a piecewise function, its computation is very simple, as opposed to solving differential equations in the case of CPGs. The research in this thesis on legged locomotion will be focused on providing a generalization of the Buehler clock approach based on max-plus-linear systems that allows for easy, fast and stable gait transitions.

1.3 Thesis outline and contributions

The outline of the thesis is given in Figure 1.4. The introduction and background on railway management systems and legged locomotion are presented in Chapter 1. In Chapter 2 the theory of max-plus algebra and max-plus-linear systems is discussed. Then the research consists of two parts: Chapter 3 and 4 on railway traffic management, and Chapter 5 on legged locomotion. The conclusions and recommendations of both parts of research are combined in Chapter 6.

The research on railway traffic management described in this thesis is based on the framework for railway traffic management given in Figure 1.2. The model for the model predictive controller is the focus of Chapter 3. First the model for the railway traffic during nominal operations, when no control actions are taken, is introduced and described as a max-plus-linear system. This model is then extended to include rescheduling actions such as reordering of trains, breaking connections, switching tracks, and splitting coupled trains, resulting in a switching max-plus-linear system. It is then shown how this model can be transformed and reduced in size. The main contributions of this chapter are the introduction of rescheduling actions for switching tracks, the formulation of the system in matrix form, the transformation of the model into its explicit form, the reduction method for the explicit form, and the modeling of freight trains.

In Chapter 4 the model predictive control approach is described and methods for improving the time needed to solve a single step of the model predictive controller are

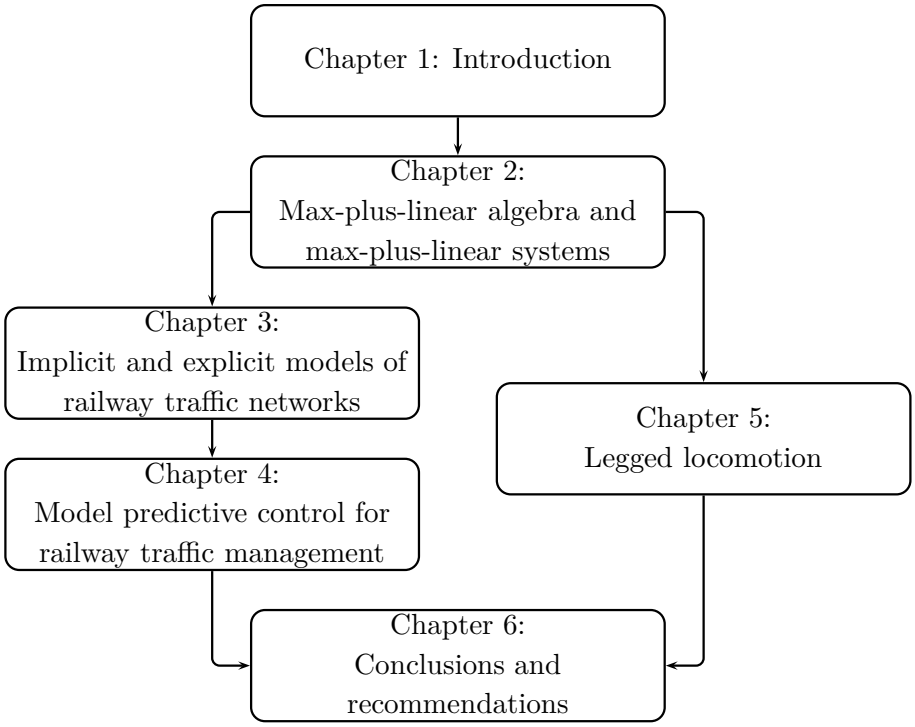

Figure 1.4: Outline of the thesis.

proposed. In a single step the model predictive controller solves the railway traffic management problem for the railway operations on the entire passenger railway network using a macroscopic model of the railway operations for a given prediction and control horizon. The control actions are limited to changing the order of trains, breaking train connections, changing the tracks trains are driving on, and breaking joined trains. Rerouting trains in and around the interlocking areas of stations is not considered. We first focus on a single step because each step needs to be solved very fast, since the controller needs to be usable during real-time operations. Because the problem is a mixed integer linear programming (MILP) problem the computation time will, in the worst case, increase exponentially with the number of binary control variables. For the model predictive controller to be usable on-line, the railway traffic management problem has to be solved fast for the network of a whole country for a long prediction and control horizon. This is especially true when the optimization has to be repeated several times with updated information because of the interaction between the model predictive controller and the trajectory and local route planning system. The main contributions of Chapter 4 are the distributed model predictive control methods that are proposed to reduce the computation time of a single step of the controller. Furthermore in Chapter 4 the different models and approaches are extensively tested in various case studies including simulations of a single step of the model predictive controller but also using a receding horizon in order to determine whether the proposed control approach results in an overall reduction in the delays or only moves the delays out of the prediction horizon.

In Chapter 5 an alternative to the common continuous time modeling approach for legged locomotion is proposed. An abstraction to represent the combinatorial nature of

the gait space for multi-legged robots into ordered sets of leg index numbers is introduced. This abstraction combined with max-plus-linear equations allows for systematic synthesis and implementation of motion controllers for multi-legged robots where gait switching is natural and the translation to continuous-time motion controllers is straightforward. The methodology presented is particularly relevant for robots with four, six, or higher numbers of legs where the number of possible gaits and gait switches becomes very large. For a large number of legs it is not obvious in which order each leg should be in swing or in stance. Most legged animals, in particular large mammals, are known to walk and run with various gaits on a daily basis, depending on the terrain or on how fast they need to move. The discrete-event framework presented in Chapter 5 enables the same behavior for multi-legged robots. The main contributions in this chapter consist of the proof of the uniqueness of the eigenvector, optimal gait switching, and the simulations of the max-plus-gait scheduler.

In Chapter 6 the contributions to the research on railway traffic management and legged locomotion are discussed and recommendations for future research are given.

Chapter 2

Max-plus-linear algebra and max-plus-linear systems

In this chapter an overview of the theory and definitions of max-plus algebra and max-plus-linear systems is given. Furthermore max-plus-linear and switching max-plus-linear systems are described. The chapter is meant as background information for those unfamiliar with max-plus algebra.

2.1 Introduction

In the sixties of the last century several researchers [17, 18, 33, 34] independently discovered that discrete-event systems in which only synchronization and no concurrency or choice occur, can be described by models using only the operators max (used to model the synchronization between events: an event can only occur as soon as all processes it depends on have finished) and + (used to model the process times: the moment a process finishes equals the moment it started plus the time the process takes to finish). These discrete-event systems are called max-plus-linear systems since they are "linear" in the max-plus algebra [3, 45]. In Section 2.2 the theory and definitions of the max-plus algebra that are needed in the rest of the thesis are explained. In Section 2.3 the max-plus system theory is explained and extended to switching max-plus-linear systems.

2.2 Max-plus algebra

The max-plus algebra is an idempotent semi-ring, consisting of the set $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$, where $\varepsilon = -\infty$, equipped with the two operators $\oplus$ and $\otimes$, which are defined as follows [3, 19, 45]:

$$a \oplus b = \max(a, b) \tag{2.1}$$

$$a \otimes b = a + b, \tag{2.2}$$

for $a, b \in \mathbb{R}_\varepsilon$. During evaluation $\otimes$ has priority over $\oplus$. Note that $a \otimes \varepsilon = \varepsilon$ for all $a \in \mathbb{R}_\varepsilon$.

For example

$$\max(a+c, b+c, d+e) = (a \otimes c) \oplus (b \otimes c) \oplus (d \otimes e).$$

For matrices these operators are defined as:

$$[A \oplus B]_{i,j} = [A]_{i,j} \oplus [B]_{i,j} = \max([A]_{i,j}, [B]_{i,j}) \tag{2.3}$$

$$[A \otimes C]_{i,j} = \bigoplus_{m=1}^{n} [A]_{i,m} \otimes [C]_{m,j} = \max_{m=1,\ldots,n} ([A]_{i,m} + [C]_{m,j}), \tag{2.4}$$

where $A, B \in \mathbb{R}_{\varepsilon}^{m \times n}$ and $C \in \mathbb{R}_{\varepsilon}^{n \times p}$.

As an example consider the matrices

$$A = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ 2 & \varepsilon & 2 \\ \varepsilon & 3 & \varepsilon \end{bmatrix}$$

$$B = \begin{bmatrix} 3 & \varepsilon & 1 \\ 4 & 2 & \varepsilon \\ 5 & \varepsilon & \varepsilon \end{bmatrix}$$

$$C = \begin{bmatrix} 2 & \varepsilon \\ 3 & \varepsilon \\ \varepsilon & 1 \end{bmatrix},$$

then

$$[A \oplus B] = \begin{bmatrix} \varepsilon \oplus 3 & \varepsilon \oplus \varepsilon & \varepsilon \oplus 1 \\ 2 \oplus 4 & \varepsilon \oplus 2 & 2 \oplus \varepsilon \\ \varepsilon \oplus 5 & 3 \oplus \varepsilon & \varepsilon \oplus \varepsilon \end{bmatrix} = \begin{bmatrix} 3 & \varepsilon & 1 \\ 4 & 2 & 2 \\ 5 & 3 & 1 \end{bmatrix}$$

$$[A \otimes C] = \begin{bmatrix} \varepsilon \otimes 2 \oplus \varepsilon \otimes 3 \oplus \varepsilon \otimes \varepsilon & \varepsilon \otimes \varepsilon \oplus \varepsilon \otimes \varepsilon \oplus \varepsilon \otimes 1 \\ 2 \otimes 2 \oplus \varepsilon \otimes 3 \oplus 2 \otimes \varepsilon & 2 \otimes \varepsilon \oplus \varepsilon \otimes \varepsilon \oplus 2 \otimes 1 \\ \varepsilon \otimes 2 \oplus 3 \otimes 3 \oplus \varepsilon \otimes \varepsilon & \varepsilon \otimes \varepsilon \oplus 3 \otimes \varepsilon \oplus \varepsilon \otimes 1 \end{bmatrix} = \begin{bmatrix} \varepsilon & \varepsilon \\ 4 & 3 \\ 6 & \varepsilon \end{bmatrix}.$$

The matrix $\mathcal{E}$ is the max-plus-algebraic zero matrix: $\mathcal{E}_{i,j} = \varepsilon$ for all i, j and denote a max-plus-algebraic zero matrix of dimension m by n as $\mathcal{E}_{m \times n}$. A max-plus diagonal matrix $\mathcal{D} = \mathrm{diag}_{\oplus}(\delta_1, \ldots, \delta_n) \in \mathbb{R}_{\varepsilon}^{n \times n}$ has elements $[\mathcal{D}]_{i,j} = \varepsilon$ for $i \neq j$ and diagonal elements $[\mathcal{D}]_{i,i} = \delta_i$ for $i = 1, \ldots, n$. A max-plus permutation matrix $T \in \mathbb{R}_{\varepsilon}^{m \times m}$ has one zero in each row and one zero in each column and ε elsewhere.

For $A, B \in \mathbb{R}_{\varepsilon}^{n \times m}$ we say that A *overcomes* B, written as $A \geq B$ if $A \oplus B = A$ (i.e., $[A]_{i,j} \geq [B]_{i,j}$ for all i, j).

Before the rest of the properties and definitions of max-plus algebra can be described, some definitions from graph theory are needed. These definitions will be described next. First the definition of a directed graph is given:

Definition 2.1 Directed graph
A directed graph $\mathcal{G}$ is defined as an ordered pair ($\mathcal{V}$,$\mathcal{A}$), where $\mathcal{V}$ is a set of vertices and $\mathcal{A}$ is a set of ordered pairs of vertices. The elements of $\mathcal{A}$ are called arcs. An arc of the form (v, v) is called a (self-)loop. □

Figure 2.1: Example graph consisting of three vertices and three arcs.

An example of a graph is given in Figure 2.1. The graph has three vertices $\mathcal{V} = \{v_1, v_2, v_3\}$ and three arcs $\mathcal{A} = \{(v_1, v_2), (v_2, v_3), (v_3, v_2)\}$.

A directed graph may contain several paths. A path is defined as:

Definition 2.2 Path in a directed graph
Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph with $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. A path p of length l is a sequence of vertices v_{i_1}, v_{i_2}, $\ldots$, $v_{i_{l+1}}$ such that $(v_{i_k}, v_{i_{k+1}}) \in \mathcal{A}$ for $k = 1, 2, \ldots, l$. We represent this path by $v_{i_1} \to v_{i_2} \to \ldots \to v_{i_{l+1}}$ and we denote the length of the path by $|p|_l = l$. Vertex v_{i_1} is the initial vertex of the path and $v_{i_{l+1}}$ is the final vertex of the path. □

The example in Figure 2.1 has several paths such as $v_1 \to v_2 \to v_3$, and $v_3 \to v_2$.

The set of all paths of length l from vertex v_{i_1} to v_{i_l} is denoted by $P(v_{i_1}, v_{i_l}; l)$. If for any two different vertices v_i, $v_j \in \mathcal{V}$ there exists a path from v_i to v_j then a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is called *strongly connected*.

Some paths may have the same initial and final vertex; such paths are called circuits:

Definition 2.3 Circuit in a directed graph
Given a path $v_{i_1} \to v_{i_2} \to \ldots \to v_{i_{l+1}}$, if $v_{i_1} = v_{i_{l+1}}$ this path is called a *circuit*.

If no vertex in the circuit appears more than once, except for the initial vertex v_{i_1} that appears exactly twice, then this circuit is called an *elementary circuit* □

In the graph of Figure 2.1 there is one elementary circuit: $v_2 \to v_3 \to v_2$. It can also be denoted by $v_3 \to v_2 \to v_3$, but both describe the same elementary circuit.

If we have a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with $\mathcal{V} = \{1, 2, \ldots, n\}$ and if we associate a real number $[A]_{i,j}$ with each arc $(j, i) \in \mathcal{A}$, then $\mathcal{G}$ is a weighted directed graph. The numerical value of $[A]_{i,j}$ denotes the weight of the arc (j, i). Note that the first subscript of $[A]_{i,j}$ corresponds to the final (and not the initial) vertex of the arc (j, i). This leads us to the next definition:

Definition 2.4 Precedence graph
Consider $A \in \mathbb{R}_\varepsilon^{n \times n}$. The precedence graph of A, denoted by $\mathcal{G}(A)$, is a weighted directed graph with vertices 1, 2, $\ldots$, n and an arc (j, i) with weight $[A]_{i,j}$ for each $[A]_{i,j} \neq \varepsilon$. □

Consider again the graph in Figure 2.1 and assign the following weights to the arcs: $(1,2)$ weighs 2, $(2,3)$ weighs 3, $(3,2)$ weighs 2, then the matrix A associated to this graph is given by:

$$A = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ 2 & \varepsilon & 2 \\ \varepsilon & 3 & \varepsilon \end{bmatrix}.$$

Let $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ and consider $\mathcal{G}(A)$. The weight $|p|_{\mathrm{w}}$ of a path $p : i_1 \to i_2 \to \ldots \to i_{l+1}$ is defined as the sum of the weights of the arcs that compose the path: $|p|_{\mathrm{w}} = [A]_{i_2,i_1} + [A]_{i_3,i_2} + \ldots + [A]_{i_{l+1},i_l} = \bigotimes_{k=1}^{l} [A]_{i_{k+1},i_k}$. The average weight of a circuit is defined as the weight of the circuit divided by the length of the circuit: $|p|_{\mathrm{w}}/|p|_{\mathrm{l}}$.

Furthermore the element $[A^{\otimes p}]_{i,j}$ is the maximum of the weights of all paths in the graph $\mathcal{G}(A)$ of length p from node j to node i:

$$[A^{\otimes p}]_{i,j} = \bigoplus_{\{v_{i_0}\to v_{i_1}\to \ldots \to v_{i_{p-1}}\to v_{i_p}\}\in P(j,i;p)} A_{i_p,i_{p-1}} \otimes A_{i_{p-1},i_{p-2}} \otimes \cdots A_{i_1,i_0}. \tag{2.5}$$

Using the above graph-theoretical definitions several max-plus definitions can be described next.

Definition 2.5 Irreducibility
A matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ is called irreducible if its precedence graph is strongly connected. □

Using the definitions of the precedence graph and circuits the following theorem can be stated:

Theorem 2.1 (Theorem 3.17 of Baccelli et al. [3])
Consider the following system of linear equations in the max-plus algebra:

$$x = A \otimes x \oplus b, \tag{2.6}$$

with $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ and $b, x \in \mathbb{R}_{\varepsilon}^{n\times 1}$. There exists a solution to this equation if there are only circuits of non-positive weight (or no circuits at all) in $\mathcal{G}(A)$ and the solution is given by

$$x = A^* \otimes b, \tag{2.7}$$

where A^* is defined as

$$A^* \coloneqq \bigoplus_{p=0}^{\infty} A^{\otimes p}. \tag{2.8}$$

If the circuits have negative weight, or there are no circuits, this solution is unique.

In some cases the infinite max-plus sum in (2.8) can be limited to a finite number:

Theorem 2.2 (Theorem 3.20 of Baccelli et al. [3])
If the precedence graph $\mathcal{G}(A)$ has no circuits of positive weight, then

$$A^* = E \oplus A \oplus A^{\otimes 2} \oplus \ldots \oplus A^{\otimes n-1},$$

where n is the dimension of A.

This means that if $A^{\otimes p}$ for $p = 1, \ldots, n$ does not contain any positive diagonal elements then the infinite sum in (2.8) can be limited to the range $p = 1, \ldots, n-1$.

Related to this theorem is the definition of a nilpotent matrix [3]:

Definition 2.6 Nilpotent Matrix
The matrix $A \in \mathbb{R}_{\varepsilon}^{n \times n}$ is called nilpotent if there exists a finite positive integer p_0 such that for all integers $p \geq p_0$ we have $A^{\otimes p} = \mathcal{E}$. □

Next the max-plus eigenvalue and eigenvector are defined:

Definition 2.7 Max-plus eigenvalue and eigenvector
Let $A \in \mathbb{R}_{\varepsilon}^{n \times n}$. If there exist a number $\lambda \in \mathbb{R}$ and a vector $v \in \mathbb{R}_{\varepsilon}^{n}$ with $v \neq \mathcal{E}_{n \times 1}$ such that

$$A \otimes v = \lambda \otimes v, \tag{2.9}$$

then λ is a max-plus eigenvalue of A and v is a corresponding max-plus eigenvector of A. □

In contrast to linear algebra, the number of max-plus eigenvalues of an n by n matrix can be less than n. In max-plus algebra every square matrix with entries in $\mathbb{R}_{\varepsilon}$ has at least one max-plus eigenvalue. If that matrix is irreducible, it has only one max-plus eigenvalue (see e.g. [3]). Similarly to an eigenvector in regular algebra, if a vector v is a max-plus eigenvector of A, then $\alpha \otimes v$ with $\alpha \in \mathbb{R}$ is also a max-plus eigenvector of A.

The max-plus eigenvalue can be interpreted in the following graph-theoretical manner.

Consider $A \in \mathbb{R}_{\varepsilon}^{n \times n}$. If λ satisfies (2.9) then there exists a circuit in $\mathcal{G}(A)$ with average weight equal to λ. If A is irreducible then λ is the maximal average weight of all elementary circuits in $\mathcal{G}(A)$ and is the unique eigenvalue of A. We denote the unique eigenvalue with $\lambda_{\max}$.

Every circuit of $\mathcal{G}(A)$ with an average weight that is equal to $\lambda_{\max}$ is called a critical circuit. The *critical graph* $\mathcal{G}_{\mathrm{c}}(A)$ of the matrix A is the set of all critical circuits.

Another max-plus algebraic property related to the eigenvector and eigenvalue is the cyclicity of a matrix. The cyclicity is defined as:

Definition 2.8 Cyclicity of a matrix
A matrix A is said to be cyclic if there exists an eigenvalue $\lambda \in \mathbb{R}$, and integers $c \in \mathbb{N}$ and $k_0 \in \mathbb{N}$ such that

$$\forall p \geq k_0 : A^{\otimes p+c} = \lambda^{\otimes c} \otimes A^{\otimes p}. \tag{2.10}$$

The smallest c that satisfies this definition is called the cyclicity of matrix A; k_0 is called the coupling time of A. □

Theorem 2.3 (Irreducibility and cyclicity [3, 45])
Any irreducible matrix A is cyclic.

2.3 Max-plus-linear systems

In the introduction of this chapter it was mentioned that discrete-event systems in which only synchronization and no concurrency or choice occur, can be modeled as max-plus-linear systems. In this section we will describe these max-plus-linear systems and extend the description to switching max-plus-linear systems.

2.3.1 Max-plus-linear systems

Some examples of discrete-event systems that can be modeled as max-plus-linear systems are manufacturing plants, railway traffic, the movement of legged robots, the scheduler for a multi-core processor. In all of these systems the behavior is defined by a sequence of activities that occur. The start and end of these activities are called the events of the discrete-event system. In the case of a manufacturing plant materials undergo several processes in a certain order to arrive at a final product. In this case the events are the start and end of the processes the materials undergo. Often the complete process from raw material till finished product is repeated resulting in a periodic behavior. This periodic event based behavior can be described by a max-plus-linear system.

Consider a (period-varying) max-plus-linear system with m inputs and l outputs, this system can be described as:

$$x(k) = \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu(k) \otimes x(k-\mu) \oplus B(k) \otimes h(k) \tag{2.11}$$

$$y(k) = C(k) \otimes x(k), \tag{2.12}$$

where k is the cycle counter, $x(k-\mu) \in \mathbb{R}_\varepsilon^{n\times 1}$ is the state vector of the system containing the times that the events occurred for the $(k-\mu)$-th time. For example the start and end of all processes for the first batch of raw materials is in $x(1)$, the start and end of all processes for the second batch of materials is in $x(2)$. The vector $h(k) \in \mathbb{R}_\varepsilon^{m\times 1}$ is the input vector of the system containing the times the inputs are available for the system for the k-th time. For the manufacturing plant $h(k)$ would contain the times the raw materials of the k-th batch are available to undergo the processes. $y(k) \in \mathbb{R}_\varepsilon^{l\times 1}$ is the output vector containing the times the system outputs started or finished. For the example of the manufacturing plants the output is usually the finished product and then $y(k)$ contains the time(s) the k-th batch of finished products is ready. $A_\mu(k) \in \mathbb{R}_\varepsilon^{n\times n}$ for $\mu \in \{0,\ldots,\mu_{\max}\}$, $B(k) \in \mathbb{R}_\varepsilon^{n\times m}$, and $C(k) \in \mathbb{R}_\varepsilon^{l\times n}$ are the system matrices containing the process times between events, and $\mu_{\max}$ is the maximum μ for which a relation exists between $x(k)$ and $x(k-\mu)$. If the system is period-varying the system matrices may be different for different k, if it is not event-varying the system matrices remain the same for all k.

The system in (2.11)–(2.12) is in the so-called implicit form since the state vector $x(k)$ not only depends on previous state vectors, but also on itself (μ starts at zero and not at one). When the state vector only depends on previous state vectors the max-plus-linear

system is said to be in its explicit form. The explicit form is given by:

$$x(k) = \bigoplus_{\mu=1}^{\mu_{\max}} \check{A}_\mu(k) \otimes x(k-\mu) \oplus \check{B}(k) \otimes h(k) \tag{2.13}$$

$$y(k) = C(k) \otimes x(k), \tag{2.14}$$

where $\check{A}_\mu(k) \in \mathbb{R}_\varepsilon^{n\times n}$, $\check{B}(k) \in \mathbb{R}_\varepsilon^{n\times m}$, and $C(k) \in \mathbb{R}_\varepsilon^{l\times n}$ are the new system matrices. Matrix $C(k)$ is the same for the implicit and explicit form.

Using Theorem 2.1 implicit max-plus-linear systems can be transformed into explicit max-plus-linear systems if there are only circuits of non-positive weight (or no circuits at all) in $\mathcal{G}(A_0)$. For the implicit system (2.11)–(2.12) this means that the precedence graph $\mathcal{G}(A_0(k))$ can only have circuits of non-positive weight or have no circuits at all. If that is the case then the system matrices of the explicit max-plus-linear system can be written as a function of the system matrices of the implicit max-plus-linear system:

$$\check{A}_\mu(k) = A_0^*(k) \otimes A_\mu(k) \text{ for } \mu = 1, \ldots, \mu_{\max}$$
$$\check{B}(k) = A_0^*(k) \otimes B(k).$$

The theory of max-plus algebra can be used to analyze the behavior of these systems. The most important are the max-plus eigenvalue and eigenvector, which can be used to determine the asymptotic behavior of the system, and the cyclicity and coupling time, which can be used to analyze the transient behavior. The coupling time determines the maximum number of cycles needed to reach the asymptotic behavior and the cyclicity c determines whether the asymptotic behavior will be determined by a single eigenvector for $c = 1$ or if it will follow a repeating pattern of l vectors (for $c = l$).

2.3.2 Switching max-plus-linear systems

Discrete-event systems that have different operating modes, for example if the manufacturing plant produces different products using the same processes but in a different order, then each mode is described by a different set of system matrices. This type of system can be described by a switching max-plus-linear (SMPL) system [85]. The mode is denoted by $\vartheta(k) \in \{1, \ldots, o\}$ for event counter k, where o is the number of modes of the system. Switching the mode of operation means the system matrices change. An implicit SMPL system is described by

$$x(k) = \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu^{\vartheta(k)}(k) \otimes x(k-\mu) \oplus B^{\vartheta(k)}(k) \otimes h(k) \tag{2.15}$$

$$y(k) = C^{\vartheta(k)}(k) \otimes x(k), \tag{2.16}$$

where $A_\mu^{\vartheta(k)}(k)$, $B^{\vartheta(k)}(k)$, $C^{\vartheta(k)}(k)$ are the system matrices for mode $\vartheta(k)$. The switching makes it possible for the behavior of the system to change, which means the order of events can change or the process times may change. In general, the mode $\vartheta(k)$ of the system is determined by a switching function:

$$\vartheta(k) = \phi(x(k-1), \ldots, x(0), \vartheta(k-1), h(k), g(k)), \tag{2.17}$$

where $\phi(\cdot) : \mathbb{R}_\varepsilon^{n\times 1} \times \mathbb{N} \times \mathbb{R}_\varepsilon^{m\times 1} \times \mathbb{R}_\varepsilon^{p\times 1} \to \mathbb{N}$ may depend on the previous state-vectors $x(k-1),\ldots,x(0)$, the previous mode $\vartheta(k-1)$, the input of the system $h(k)$, and a control vector $g(k) \in \mathbb{R}_\varepsilon^{p\times 1}$.

For the implicit SMPL model it also applies that it can be transformed into the explicit form, but only if $A_0^{\vartheta(k)}(k)$ satisfies the conditions of Theorem 2.1 for every mode.

A special type of SMPL systems exists where the system matrices of all different modes can be combined into a single set of system matrices $A_\mu(u(k-\mu),k) \in \mathbb{R}_\varepsilon^{n\times n}$ for $\mu = 0,\ldots,\mu_{\max}$, $B(u(k),k) \in \mathbb{R}_\varepsilon^{n\times m}$, and $C(u(k),k) \in \mathbb{R}_\varepsilon^{l\times n}$ that depends on a set of max-plus binary decision variables $u(k)$, where $u_i(k)$ is defined as $u_i(k) \in \{0,\varepsilon\}$ and define the adjoint $\overline{u_i(k)} \in \{0,\varepsilon\}$ as:

$$\overline{u_i(k)} = \begin{cases} \varepsilon & \text{if } u_i(k) = 0 \\ 0 & \text{if } u_i(k) = \varepsilon. \end{cases} \tag{2.18}$$

For this special type of SMPL system each element of the system matrices can be described as a max-plus-linear equation of the form

$$f = \bigoplus_j \bigotimes \tau_j(k) \otimes \bigotimes_i u_i(k),$$

where $\tau_j(k) \in \mathbb{R}_\varepsilon$ is a process time and $u_i(k) \in \{0,\varepsilon\}$ is a max-plus binary variable. In this case the implicit SMPL system can be described as:

$$x(k) = \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu(u(k),k) \otimes x(k-\mu) \oplus B(u(k),k) \otimes h(k) \tag{2.19}$$

$$y(k) = C(u(k),k) \otimes x(k). \tag{2.20}$$

This type of SMPL systems can also be transformed into its explicit form, but only if for all combinations of binary variables the matrix $A_0(u(k),k)$ satisfies the conditions of Theorem 2.1. In special cases this requirement can be relaxed, but this depends on the application as we will show in the next chapter.

2.4 Summary

In this chapter the basics of the max-plus algebra were explained. The max-plus algebra is an idempotent semi-ring, consisting of the set $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$, where $\varepsilon = -\infty$, equipped with the two operators $\oplus$ and $\otimes$. Furthermore it was explained that discrete-event systems in which only synchronization and no concurrency or choice occur, can be modeled as max-plus-linear systems. The mathematical description of max-plus-linear systems was given. It was shown how the implicit model description is turned into the explicit model description. For those discrete-event systems that have different operating modes an extension to max-plus-linear systems was described, namely switching max-plus-linear systems. The switching max-plus-linear system can switch between the different operating modes using a switching function. Conditions were given under which the implicit switching max-plus-linear system can be rewritten into its explicit form.

Chapter 3

Implicit and explicit models of the railway traffic networks

An implicit switching max-plus-linear model of passenger railway traffic, for use in on-line railway traffic management, is presented in this chapter. Further more it is shown how this model can be converted into its explicit form and reduced in complexity. The presented model is extended to also model freight trains. Parts of this chapter have been published in [57]. The main contributions of this chapter are the formulation of the system in matrix form in Sections 3.2.3 and 3.3, the introduction of rescheduling actions for switching tracks in 3.3.4, the transformation of the model into its explicit form in Section 3.4, the reduction method for the explicit form in Section 3.5, and the modeling of freight trains in Section 3.6.

3.1 Introduction

In this chapter a model of railway traffic networks for on-line railway traffic management is proposed. In the work of Braker [5, 6] it has been shown that a macroscopic model of a railway network, with a fixed routing schedule and fixed connections, can be described as a max-plus-linear model. A system that can be described by a max-plus-linear model can be characterized as a discrete-event system in which only synchronization occurs, but no concurrency or choice [3]. Braker [5, 6] models only the departures of the trains and no dispatching actions can be made. de Vries et al. [27] and Heidergott and Vries [44] extend the max-plus-linear model to include dispatching actions that allow the cancelation of connections. In the work of Goverde [37] and Goverde [38] the arrival events were modeled explicitly for the first time. Furthermore Goverde [37] and Goverde [38] introduce the concepts of slack time and recovery matrix to analyze the stability of a given timetable. In the work of Goverde [37] and Goverde [39] an algorithm to determine the delay propagation through the network is developed and further analysis of timetable stability is done. Furthermore van den Boom et al. [86] extend the work of de Vries et al. [27] and Heidergott and Vries [44] by introducing a method to reorder the trains on the tracks and determine the optimal schedule by solving a MILP problem.

We continue the work of van den Boom et al. [86] by extending the control actions to allow the trains to switch tracks when there are multiple tracks between stations. Furthermore a new max-plus matrix formulation is introduced. The matrix formulation is used to convert the model into its explicit form. For that explicit form a reduction method is proposed. Finally the modeling of the railway traffic is extended with extra constraints for freight trains. We only model the stations and junctions where the order of the trains can be changed. These stations and junctions are modeled as single nodes with infinite capacity. Stations where the train order cannot be changed are seen as part of the tracks. Each track between stations is modeled as a single link. The movement of the trains over the tracks and the operational constraints on the trains are modeled using constraints. These constraints are explained for the nominal operation in Section 3.2. Furthermore it is shown in Section 3.2 how the model can be written as a max-plus-linear one using the theory from Chapter 2. In Section 3.3 the max-plus-linear model is extended to be able to model rescheduling actions resulting in a switching max-plus-linear model. Section 3.4 shows how the switching max-plus-linear model can be transformed into its explicit form. In Section 3.5 a method is introduced to reduce the number of constraints in the explicit switching max-plus-linear model. Section 3.6 describes how freight trains can be modeled in this framework. In Section 3.7 the chapter is summarized.

3.2 Nominal operation

The nominal operation of the railway network is modeled as a cyclic discrete-event system, with cycle counter k. The events of the discrete-event system are the arrival and departure events of the trains at the stations and junctions.

The combination of the following actions: a train departing from a station, traversing a track, and arriving at the next station, is called a train run. Each train run has an index i, and an associated departure time d_i and arrival time a_i. A set of train runs, modeling the same 'physical' train, will be called a line. During nominal operation, the railway traffic operates according to the nominal timetable: the trains follow their predetermined routes, the order in which the trains depart and arrive at stations is fixed, all connections are maintained and there are no delays in the network. The operation of the railway network can be described as a set of train runs connected to each other through various constraints.

3.2.1 Constraints connecting the train runs

There are six different constraints connecting the trains:

- Running time constraints
- Continuity constraints
- Timetable constraints
- Headway constraints

- Coupling constraints
- Connection constraints

Next all of these constraints are described in more detail for the nominal operation.

Running time constraints

The relation between the arrival time and departure time of a train run can be described by a *running time* constraint. In the rest of this chapter we will simply refer to a 'train run' as a 'train'. A running time constraint is defined such that the arrival and departure of a train belong to the same cycle. This was done to simplify the structure of the system matrices as described in the next section of this chapter. If a running time is much longer than the period of the timetable it is split up into multiple connected train runs.

This results in the following definition for the running time constraint for train i in cycle k:

$$a_i(k) \geq d_i(k) + \tau_{\mathrm{r},i}(k), \tag{3.1}$$

where $\tau_{\mathrm{r},i}(k)$ is the *running time*, i.e. the time the train needs to traverse the track, for train run i in cycle k. Note that process times may vary each cycle, even during nominal operation, e.g. due to a changing number of passengers and different rolling stock, and therefore the running times depend on k.

Because of the way the running time constraints are defined, not all arrival and departure times in the same period of the timetable will be in the same cycle of the model. As a result multiple cycles may need to be considered when determining the arrival and departure times in a given time interval. The events in the cycle are all departure events of all trains for one period of the timetable and the arrival events associated to those departure events, not the arrival events that occur in that same period of the timetable.

Continuity constraints

A continuity constraint connects two trains of the same line to each other, e.g. a 'physical' train driving from one station to the next and then continuing on to a third station. This also includes trains turning or changing lines at their end station. This can be modeled by considering train i and its predecessor p_i. Let train p_i model the 'physical' train driving from the first to the second station and let train i model the 'physical' train driving from the second to the third station. Train i can then only start some time after train p_i has arrived:

$$d_i(k) \geq a_{p_i}(k - \mu_{i,p_i}) + \tau_{\mathrm{d},i,p_i}(k), \tag{3.2}$$

where $\mu_{i,p_i} = 0$ if train p_i in cycle k continues as train i in cycle k and $\mu_{i,p_i} = \alpha$ if train p_i in cycle $k - \alpha$ continues as train i in cycle k, and $\tau_{\mathrm{d},i,p_i}(k)$ is the *dwell time*, i.e. the time the train waits at the station for passengers to board and alight.

Timetable constraints

Since the passenger railways operate according to a timetable, none of the trains are allowed to depart before their scheduled departure times and in some cases they may not arrive before their scheduled arrival times either. This requirement can be modeled by

adding *timetable* constraints:

$$d_i(k) \geq r_{\mathrm{d},i}(k) \tag{3.3}$$
$$a_i(k) \geq r_{\mathrm{a},i}(k), \tag{3.4}$$

where $r_{\mathrm{d},i}(k)$ and $r_{\mathrm{a},i}(k)$ are the scheduled departure and arrival time of train i in cycle k. Note that for a cyclic timetable it holds that $r_{\mathrm{d},i}(k) = r_{\mathrm{d},i}(0) + kT$ and $r_{\mathrm{a},i}(k) = r_{\mathrm{a},i}(0) + kT$, where T is the period of the timetable/footnoteBy definition of the running time constraints the scheduled departure times of cycle k are in $[(k-1)T, kT)$, but the scheduled arrival times may not be.. In many countries trains are allowed to arrive before their scheduled arrival time; in that case the timetable constraint on the arrival time, as in (3.4), should be left out for those trains. For junctions there are no scheduled arrival or departure times the trains should adhere to, so for those events no timetable constraints are defined.

Headway constraints

Headway constraints define the order in which trains traverse tracks and they indirectly define the minimum distance between trains. This is done by relating the arrival and departure times of one train to the arrival and departure time of the other trains traversing the same track. The headway times can be chosen such that, as long as there are no unexpected delays on the tracks, none of the trains run into a yellow or red signal and have to break, or norms defined by the industry can be used. If several trains traverse a track in the same direction, then for train i set $\mathcal{H}_i$ is defined as the set of trains that start on the track before train i and traverse the track in the same direction during nominal operation. The headway constraints for train i for the trains traversing the track in the same direction are:

$$d_i(k) \geq d_l(k - \mu_{i,l}) + \tau_{\mathrm{h,d},i,l}(k) \tag{3.5}$$
$$a_i(k) \geq a_l(k - \mu_{i,l}) + \tau_{\mathrm{h,a},i,l}(k), \tag{3.6}$$

for each $l \in \mathcal{H}_i$, where $\tau_{\mathrm{h,d},i,l}(k)$ is the *headway time* for departures, i.e. the time needed between the departure of train l in cycle $k - \mu_{i,l}$ and the departure of train i in cycle k, $\tau_{\mathrm{h,a},i,l}(k)$ is the headway time needed between the arrival of train l in cycle $k - \mu_{i,l}$ and the arrival of train i in cycle k, and where $\mu_{i,l}$ is defined in the same way as for (3.2).

If trains traverse the track in the opposite direction, then for train run i, the set $\mathcal{S}_i$ is defined as the set of trains that start on the same track before train i and traverse the track in opposite direction during nominal operation. The headway constraints for train i for the trains traversing the track in the opposite direction are:

$$d_i(k) \geq a_m(k - \mu_{i,m}) + \tau_{\mathrm{s},i,m}(k), \tag{3.7}$$

for each $m \in \mathcal{S}_i$, where $\tau_{\mathrm{s},i,m}(k)$ is the *separation time*, i.e. the time the train i in cycle k must wait before it can enter the track after train m in cycle $k - \mu_{i,m}$ has left the track and where $\mu_{i,m}$ is defined in the same way as for (3.2).

Coupling constraints

At some stations two 'physical' trains are coupled and continue as a single 'physical' train; this is modeled by *coupling* constraints. In the literature it is common to model the single 'physical' train with a single train run in the model. In our model the two trains are modeled separately, even when they continue as a single 'physical' train, and connected to each other through coupling constraints. This allows us to decide if the trains should be coupled or if they should continue as two separate 'physical' trains by changing the coupling constraints. This will be shown in the Section 3.3.

The coupling constraints ensure that the arrival and departure times of the two 'physical' trains are the same. Consider train i and let train o_i be the train to which train i should be coupled to during nominal operation. For these trains the coupling constraints are:

$$d_i(k) = d_{o_i}(k) \tag{3.8}$$
$$a_i(k) = a_{o_i}(k). \tag{3.9}$$

Here it is assumed both trains are in the same cycle, since they have the same departure and arrival times. These two equality constraints can be written as four inequality constraints:

$$d_i(k) \geq d_{o_i}(k) \tag{3.10}$$
$$d_{o_i}(k) \geq d_i(k) \tag{3.11}$$
$$a_i(k) \geq a_{o_i}(k) \tag{3.12}$$
$$a_{o_i}(k) \geq a_i(k). \tag{3.13}$$

These four inequality constraints are used instead of the two equality constraints. This is because in the perturbed operation (see Section 3.3) it will be possible to cancel the coupling if one of the trains is delayed, and using these inequalities instead of equalities makes it easier to support that modification.

Connection constraints

At some stations passengers can transfer to another train. Transfers that are guaranteed, are modeled by *connection* constraints. Connection constraints ensure that passengers can change trains at stations by defining a relation between the departure time of one train and the arrival time of the train from which the passengers transfer. Define $\mathcal{C}_i$ as the set of train runs, train i has to give a connection to during nominal operation. Then the connection constraints for train i are defined as:

$$d_i(k) \geq a_e(k - \mu_{i,e}) + \tau_{\mathrm{c},i,e}(k), \tag{3.14}$$

for each $e \in \mathcal{C}_i$, and where $\tau_{\mathrm{c},i,e}(k)$ is the *connection time*, i.e. the time needed for the passengers to transfer from train e in cycle $k - \mu_{i,e}$ to train i in cycle k.

3.2.2 Max-plus-linear model

If all constraints are satisfied, the trains can depart and arrive without running into any conflict with other trains. Therefore, we assume that all trains depart and arrive as soon

as all constraints are satisfied. Then the constraints for train i can be written using two equations, one for the arrival and one for the departure time:

$$\begin{aligned} d_i(k) = \max\Bigg(& a_{p_i}(k-\mu_{i,p_i}) + \tau_{\mathrm{d},i,\mathrm{p}_i}(k), \\ & \max_{l\in\mathcal{H}_i}\Big(d_l(k-\mu_{i,l}) + \tau_{\mathrm{h,d},i,l}(k)\Big), \\ & \max_{m\in\mathcal{S}_i}\Big(a_m(k-\mu_{i,m}) + \tau_{\mathrm{s},i,m}(k)\Big), \\ & \max_{e\in\mathcal{C}_i}\Big(a_e(k-\mu_{i,e}) + \tau_{\mathrm{c},i,e}(k)\Big), \\ & d_{o_i}(k), r_{\mathrm{d},i}(k)\Bigg) \end{aligned} \tag{3.15}$$

$$\begin{aligned} a_i(k) = \max\Bigg(& \max_{l\in\mathcal{H}_i}\Big(a_l(k-\mu_{i,l}) + \tau_{\mathrm{h,a},i,l}(k)\Big), \\ & d_i(k) + \tau_{\mathrm{r},i}(k), a_{o_i}(k), r_{\mathrm{a},i}(k)\Bigg). \end{aligned} \tag{3.16}$$

Note that in an undisturbed, well-defined time schedule the terms $r_{\mathrm{d},i}(k)$ and $r_{\mathrm{a},i}(k)$ in (3.15) and (3.16) respectively will be the largest. However, if one of the trains p_i, l, m, e or o_i has a delay, due to unforeseen circumstances (an incident, a late departure, etc.), then the corresponding term can become larger than the others and train i will depart later than the scheduled departure time $r_{\mathrm{d},i}(k)$ and will therefore be delayed as well.

Now let us consider a network with n 'trains' and define the vectors

$$x(k) = \begin{bmatrix} d_1(k) \\ \vdots \\ d_n(k) \\ a_1(k) \\ \vdots \\ a_n(k) \end{bmatrix} \in \mathbb{R}_\varepsilon^{2n}, \qquad r(k) = \begin{bmatrix} r_{\mathrm{d},1}(k) \\ \vdots \\ r_{\mathrm{d},n}(k) \\ r_{\mathrm{a},1}(k) \\ \vdots \\ r_{\mathrm{a},n}(k) \end{bmatrix} \in \mathbb{R}_\varepsilon^{2n}.$$

By defining appropriate matrices $A_\mu(k) \in \mathbb{R}_\varepsilon^{2n\times 2n}$ for $\mu = 0, 1, \ldots, \mu_{\max}$, where $\mu_{\max} = \max_{i,j} \mu_{i,j}$, (3.15) and (3.16) can be rewritten as

$$\begin{aligned} x_i(k) = \max\Big(& \max_j(x_j(k) + [A_0(k)]_{i,j}), \max_j(x_j(k-1) + [A_1(k)]_{i,j}), \ldots, \\ & \max_j(x_j(k-\mu_{\max}) + [A_{\mu_{\max}}(k)]_{i,j}), r_i(k)\Big), \end{aligned} \tag{3.17}$$

where $[A_\mu(k)]_{i,j}$ is the (i,j)th entry of $A_\mu(k)$[1].

Using the max-plus algebra explained in Chapter 2, (3.17) can be written as a max-plus-linear equation:

$$x_i(k) = r_i(k) \oplus \bigoplus_{\mu=0}^{\mu_{\max}} \bigoplus_{j=1}^{2n} [A_\mu(k)]_{i,j} \otimes x_j(k-\mu). \tag{3.18}$$

[1]The matrices $A_\mu(k)$ can be completed by adding $[A_\mu(k)]_{i,j} = -\infty$ for all combinations (μ, i, j, k) that do not appear in (3.17).

By determining this max-plus-linear equation for all $x_i(k)$ the model can be written as a max-plus-linear (MPL) model defined as:

$$x(k) = r(k) \oplus \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu(k) \otimes x(k-\mu), \tag{3.19}$$

which is an MPL model of a railway network with a fixed routing schedule and fixed connections, such as the models of Braker [5] and Goverde [39]. The current cycle of the MPL model is denoted by k. For all preceding cycles $k-n$, $n \in \mathbb{N}$, all event times are assumed to be known, fixed, and in the past. That means that cycle k is the first cycle for which event times are unknown and events still have to occur.

3.2.3 System matrices for the nominal operation

From (3.19) it is clear that the dynamics of the railway system are described by the matrices $A_\mu(k)$, $\mu = 0, \ldots, \mu_{\max}$. In the remainder of this section (k) will be omitted from the notation of the $A_\mu(k)$ matrices to improve the readability. In this section we will study the structure of the matrices A_μ for the nominal operation. It can be verified that the matrices A_μ, $\mu = 0, \ldots, \mu_{\max}$ can be written as

$$A_\mu = \begin{bmatrix} A_{\mu,4,\mathrm{d}} \oplus A_{\mu,6,\mathrm{d}} & A_{\mu,2} \oplus A_{\mu,3} \oplus A_{\mu,5} \\ A_{\mu,1} & A_{\mu,4,\mathrm{a}} \oplus A_{\mu,6,\mathrm{a}} \end{bmatrix}, \tag{3.20}$$

with $A_{\mu,1}$, $A_{\mu,2}$, $A_{\mu,3}$, $A_{\mu,4,\mathrm{d}}$, $A_{\mu,4,\mathrm{a}}$, $A_{\mu,5}$, $A_{\mu,6,\mathrm{d}}$, $A_{\mu,6,\mathrm{a}} \in \mathbb{R}_\varepsilon^{n \times n}$. The structure of these six matrices will now be discussed in detail.

The running time matrix

The running time matrix, denoted by $A_{\mu,1}$, represents the running time constraints. Since by definition the arrival and departure of a train belong to the same cycle, $A_{\mu,1} = \mathcal{E}$ for all $\mu \neq 0$. As a result, $A_{\mu,1}$ has the following structure:

$$A_{\mu,1} = \begin{cases} \mathrm{diag}_\oplus(\tau_{\mathrm{r},1}(k), \quad \tau_{\mathrm{r},2}(k), \quad \cdots, \quad \tau_{\mathrm{r},n}(k)) & \text{for } \mu = 0 \\ \mathcal{E} & \text{for } \mu \neq 0. \end{cases} \tag{3.21}$$

The dwell time matrix

The dwell time matrix, denoted by $A_{\mu,2}$, represents the continuity constraints. The structure of this matrix is defined as follows:

$$[A_{\mu,2}]_{i,j} = \begin{cases} \tau_{\mathrm{d},i,p_i}(k) & \text{if } j = p_i \text{ and } \mu = \mu_{i,p_i} \\ \varepsilon & \text{else.} \end{cases} \tag{3.22}$$

Let n_{L} be the number of lines in the network, let $n_{\mathrm{l},m}$ be the number of trains on line m, $m = 1, \ldots, n_{\mathrm{L}}$, so $n_{\mathrm{l},1} + n_{\mathrm{l},2} + \ldots + n_{\mathrm{l},n_{\mathrm{L}}} = n$, and let $L_m \in \mathbb{R}^{n_{\mathrm{l},m}}$ be a vector containing the indices of the trains of line m, with $L_{m,i}$ the ith element of vector L_m. Define a max-plus permutation matrix $E_{\mathrm{dwell}} \in \mathbb{R}_\varepsilon^{n \times n}$ that orders the rows and columns of $A_{\mu,2}$, such that the associated event times are ordered by line[2] and per line by the scheduled departure

[2]Recall that a line is defined as the set of train runs modeling the same 'physical' train.

times:

$$A_{\mu,2} = E_{\text{dwell}} \otimes \begin{bmatrix} \hat{A}_{\mu,2,1} & \mathcal{E} & \cdots & \mathcal{E} \\ \mathcal{E} & \hat{A}_{\mu,2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathcal{E} \\ \mathcal{E} & \cdots & \mathcal{E} & \hat{A}_{\mu,2,n_L} \end{bmatrix} \otimes E_{\text{dwell}}^{\top}, \tag{3.23}$$

where $\hat{A}_{\mu,2,m} \in \mathbb{R}^{n_{\text{l},m} \times n_{\text{l},m}}$ can be described as:

$$\hat{A}_{\mu,2,m} = \begin{bmatrix} \varepsilon & \cdots & \cdots & \varepsilon & \hat{\tau}_{\text{d},m,\mu,1,n_{\text{l},m}}(k) \\ \hat{\tau}_{\text{d},m,\mu,2,1}(k) & \ddots & \ddots & \vdots & \varepsilon \\ \varepsilon & \hat{\tau}_{\text{d},m,\mu,3,2}(k) & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \varepsilon & \vdots \\ \varepsilon & \cdots & \varepsilon & \hat{\tau}_{\text{d},m,\mu,n_{\text{l},m},n_{\text{l},m}-1}(k) & \varepsilon \end{bmatrix}, \tag{3.24}$$

where

$$\hat{\tau}_{\text{d},m,\mu,i,j}(k) = \begin{cases} \tau_{\text{d},L_{m,i},L_{m,j}}(k) & \text{if } L_{m,j} = p_{L_{m,i}} \text{ and } \mu = \mu_{L_{m,i},L_{m,j}} \\ \varepsilon & \text{else.} \end{cases}$$

If $L_{m,i} = p_{L_{m,j}}$ then train $L_{m,i}$ is the predecessor of train $L_{m,j}$ – in other words, train $L_{m,i}$ continues as train $L_{m,j}$ – and there should be a dwell time between the arrival of train $L_{m,j}$ and the departure of train $L_{m,i}$.

The connection matrix

The connection matrix represents the connection constraints and is denoted by $A_{\mu,3}$. This matrix is structured as follows:

$$[A_{\mu,3}]_{i,j} = \begin{cases} \tau_{\text{c},i,j}(k) & \text{if } j \in \mathcal{C}_i \text{ and } \mu = \mu_{i,j} \\ \varepsilon & \text{else.} \end{cases} \tag{3.25}$$

The headway matrices

The matrices $A_{\mu,4,\text{d}}$ and $A_{\mu,4,\text{a}}$ represent the headway constraints for trains in the same direction on the same track.

Let n_{T} be the number of tracks in the network, let $n_{\text{t},m}$ be the number of trains on track m, $m = 1, \ldots, n_{\text{T}}$, so $n_{\text{t},1} + n_{\text{t},2} + \ldots + n_{\text{t},n_{\text{T}}} = n$, and let $T_m \in \mathbb{R}^{n_{\text{t},m}}$ be a vector containing the indices of the trains on track m, ordered according to the timetable. Define a max-plus permutation matrix $E_{\text{t}} \in \mathbb{R}_{\varepsilon}^{n \times n}$ that reorders the half of the state vector $x(k)$ containing the departure events $d_1(k)$ up to $d_n(k)$ such that the event times are ordered per track and for each track the events are ordered according to the scheduled departure times. The matrices $A_{\mu,4,\text{d}}$ can then be defined as:

$$A_{\mu,4,\text{d}} = E_{\text{t}} \otimes \begin{bmatrix} \hat{A}_{\mu,4,\text{d},1} & \mathcal{E} & \cdots & \mathcal{E} \\ \mathcal{E} & \hat{A}_{\mu,4,\text{d},2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathcal{E} \\ \mathcal{E} & \cdots & \mathcal{E} & \hat{A}_{\mu,4,\text{d},n_{\text{T}}} \end{bmatrix} \otimes E_{\text{t}}^{\top}, \tag{3.26}$$

where $\hat{A}_{\mu,4,\mathrm{d},m} \in \mathbb{R}^{n_{\mathrm{t},m} \times n_{\mathrm{t},m}}$ and where for $\mu = 0$ we have

$$\hat{A}_{0,4,\mathrm{d},m} = \begin{bmatrix} \varepsilon & \hat{\tau}_{\mathrm{h,d},m,0,1,2}(k) & \dots & \hat{\tau}_{\mathrm{h,d},m,0,1,n_{\mathrm{t},m}}(k) \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \hat{\tau}_{\mathrm{h,d},m,0,n_{\mathrm{t},m}-1,n_{\mathrm{t},m}}(k) \\ \varepsilon & \dots & \dots & \varepsilon \end{bmatrix}, \tag{3.27}$$

and for $\mu = 1, \dots, \mu_{\max}$

$$[\hat{A}_{\mu,4,\mathrm{d},m}]_{i,j} = \hat{\tau}_{\mathrm{h,d},m,\mu,i,j}(k), \tag{3.28}$$

with

$$\hat{\tau}_{\mathrm{h,d},m,\mu,i,j}(k) = \begin{cases} \tau_{\mathrm{h,d},T_{m,i},T_{m,j}}(k) & \text{if } T_{m,j} \in \mathcal{H}_{T_{m,i}}, \text{ and } \mu = \mu_{T_{m,i},T_{m,j}} \\ \varepsilon & \text{else.} \end{cases} \tag{3.29}$$

For $A_{\mu,4,\mathrm{a}}$, the structure of the matrix is the same as for $A_{\mu,4,\mathrm{d}}$, the only differences being that in (3.26)- (3.29) $\hat{A}_{\mu,4,\mathrm{d},m}$ is replaced by $\hat{A}_{\mu,4,\mathrm{a},m}$, $\hat{\tau}_{\mathrm{h,d},m,\mu,i,j}(k)$ is replaced by $\hat{\tau}_{\mathrm{h,a},m,\mu,i,j}(k)$, and $\tau_{\mathrm{h,d},T_{m,i},T_{m,j}}(k)$ is replaced by $\tau_{\mathrm{h,a},T_{m,i},T_{m,j}}(k)$.

As an example consider the headway constraints of three trains running over the same track in the same cycle (so $\mu = 0$ for the headway constraints). Let us assume that the headway times between all three trains are 3 minutes, then

$$\begin{aligned} T_1 &= \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^\top \\ \mathcal{H}_1 &= \emptyset \\ \mathcal{H}_2 &= \{1\} \\ \mathcal{H}_3 &= \{1, 2\}, \end{aligned}$$

and the headway constraints between the departures can be determined with (3.42) resulting in:

$$A_{0,4,\mathrm{d}}(u(k), k) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon \\ 3 & 3 & \varepsilon \end{bmatrix}.$$

These headway constraints ensure that train 2 departs at least 3 minutes after train 1, and train 3 departs at least 3 minutes after train 2. There is an extra constraint that ensures train 3 departs at least 3 minutes after train 1, but in this case that constraint is not needed, since the other two constraints already ensure that train 3 departs at least 6 minutes after train 1. This constraint is only there in case the order of trains is changed during the disturbances and train 3 is driving directly behind train 1.

The separation matrix

The separation matrix is denoted by $A_{\mu,5}$, and represents the headway constraints for trains driving over the same track in the opposite direction. Using the same permutation

matrix E_{t} as for the headway matrices, $A_{\mu,5}$ can be written as:

$$A_{\mu,5} = E_{\mathrm{t}} \otimes \begin{bmatrix} \hat{A}_{\mu,5,1} & \mathcal{E} & \cdots & \mathcal{E} \\ \mathcal{E} & \hat{A}_{\mu,5,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathcal{E} \\ \mathcal{E} & \cdots & \mathcal{E} & \hat{A}_{\mu,5,n_{\mathrm{T}}} \end{bmatrix} \otimes E_{\mathrm{t}}^{\top}, \tag{3.30}$$

where $\hat{A}_{\mu,5,m} \in \mathbb{R}^{n_{\mathrm{t},m} \times n_{\mathrm{t},m}}$ and can be described as:

$$[\hat{A}_{\mu,5,m}]_{i,j} = \begin{cases} \tau_{\mathrm{s},T_{m,i},T_{m,j}}(k) & \text{if } T_{m,j} \in \mathcal{S}_{T_{m,i}} \text{ and } \mu = \mu_{T_{m,i},T_{m,j}} \\ \varepsilon & \text{else.} \end{cases} \tag{3.31}$$

The coupling matrices

The coupling matrices, denoted by $A_{\mu,6,\mathrm{d}}$ and $A_{\mu,6,\mathrm{a}}$, define which trains are coupled into one single train. The matrices $A_{\mu,6,\mathrm{d}}$ and $A_{\mu,6,\mathrm{a}}$ have the following structure:

$$[A_{\mu,6,\mathrm{d}}]_{i,j} = \begin{cases} 0 & \text{if } j = o_i \text{ and } \mu = 0 \\ 0 & \text{if } i = o_j \text{ and } \mu = 0 \\ \varepsilon & \text{else.} \end{cases} \tag{3.32}$$

Furthermore, $A_{\mu,6,\mathrm{a}} = A_{\mu,6,\mathrm{d}}$. The reason for defining two matrices that are identical is that during perturbed operation these matrices may differ: they will depend on max-plus binary variables and different process times and for many combinations of the max-plus binary variables they will be different.

3.3 Perturbed operation

The model described in the previous section has a static structure: the order of the trains on the tracks is fixed, connections cannot be broken, coupled trains cannot be decoupled, process times are fixed, and when there are multiple tracks trains can use, they cannot change tracks. In this section the abilities to change the order of the trains, break connections, decouple trains, and change tracks will be added step by step. These abilities are added by extending the model of the previous section to a switching max-plus-linear (SMPL) model by modifying constraints and adding max-plus binary variables, as defined in (2.18), to the constraints. It is called an SMPL model, since it can switch between behaviors (train orders, track choices, broken connections). The SMPL model can be described as:

$$x(k) = r(k) \oplus \bigoplus_{\mu=0}^{\mu_{\max}} A_{\mu}(k, u(k-\mu)) \otimes x(k-\mu) \oplus \bigoplus_{\mu=-\mu_{\max}}^{-1} A_{\mu}(k, u(k)) \otimes x(k-\mu), \tag{3.33}$$

where $u(k-\mu)$ is a vector containing all max-plus binary variables used in the rest of this section. The elements of A_{μ} contain the modified constraints with max-plus binary variables from u. The first max-plus sum for $\mu = 0, 1, \ldots, \mu_{\max}$ contains the part of the

model that describes the dependency of the event times of the current cycle to the event times of previous cycles. Since the event times of previous cycles are known, fixed and in the past the max-plus binary variables relating these event times to event times of the current cycle are also fixed, known and in the past. As a result A_μ depends on $u(k-\mu)$ when $\mu \geq 0$ and not on $u(k)$. By adding the ability to change the order of the trains, break connections, decouple trains, and change tracks new constraints are introduced between trains in the current cycle k and future cycles $k-\mu$, for $\mu < 0$. For example if a train in the current cycle is delayed a lot and is delaying a train in the next cycle we should be able to change the order of these trains. This involves constraints between an event in cycle k and in cycle $k+1$. By adding the second max-plus sum from $-\mu_{\max}$ to -1 these constraints can also be modeled. Since the max-plus binary variables in the second max-plus sum are taken during the kth cycle, A_μ in the second max-plus sum depends on $u(k)$.

For $A_\mu(k, u(k-\mu))$, with $\mu \in \{0, \ldots, \mu_{\max}\}$, and their sub matrices, the argument is always $(k, u(k-\mu))$ and will therefore be omitted in the remainder of this section. For $A_\mu(k, u(k))$, with $\mu \in \{-\mu_{\max}, \ldots, -1\}$, and their sub matrices, the argument is always $(k, u(k))$ and will also be omitted in the sequel.

3.3.1 Changing the order of trains

To change the order of trains on a track, the headway constraints need to be manipulated. As an example consider two trains running over the same track and in the same direction. These trains are described by train i in cycle k and l in cycle $k-\mu_{i,l}$. If the order of trains is 'l before i', then headway constraints (3.5) and (3.6) define this order. If the order is 'i before l' then the following headway constraints should replace (3.5) and (3.6):

$$d_l(k-\mu_{i,l}) \geq d_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \tag{3.34}$$
$$a_l(k-\mu_{i,l}) \geq a_i(k) \otimes \tau_{\mathrm{h,a},l,i}(k-\mu_{i,l}). \tag{3.35}$$

To be able to change the order of trains, it is necessary to be able to turn headway constraints on and off. This can be done by multiplying the right-hand side of (3.5) and (3.6) with binary variable $u_{i,l}(k-\mu_{i,l}) \in \{0, \varepsilon\}$, and that of (3.34) and (3.35) with the adjoint binary variable $\overline{u_{i,l}(k-\mu_{i,l})} \in \{0, \varepsilon\}$. Then we obtain:

$$d_i(k) \geq d_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,d},i,l}(k) \otimes u_{i,l}(k-\mu_{i,l}) \tag{3.36}$$
$$a_i(k) \geq a_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,a},i,l}(k) \otimes u_{i,l}(k-\mu_{i,l}) \tag{3.37}$$
$$d_l(k-\mu_{i,l}) \geq d_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \otimes \overline{u_{i,l}(k-\mu_{i,l})} \tag{3.38}$$
$$a_l(k-\mu_{i,l}) \geq a_i(k) \otimes \tau_{\mathrm{h,a},l,i}(k-\mu_{i,l}) \otimes \overline{u_{i,l}(k-\mu_{i,l})}. \tag{3.39}$$

To illustrate the effect of the max-plus binary variables consider the case that $u_{il}(k-\mu_{i,l}) = 0$ in (3.36)-(3.39), then $\overline{u_{il}(k-\mu_{i,l})} = \varepsilon$ and the equations become:

$$\begin{aligned}
d_i(k) &\geq d_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,d},i,l}(k) \otimes 0 = d_l(k) \otimes \tau_{\mathrm{h,d},i,l}(k) \\
a_i(k) &\geq a_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,a},i,l}(k) \otimes 0 = a_l(k) \otimes \tau_{\mathrm{h,a},i,l}(k) \\
d_l(k-\mu_{i,l}) &\geq d_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \otimes \varepsilon = \varepsilon \\
a_l(k-\mu_{i,l}) &\geq a_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \otimes \varepsilon = \varepsilon.
\end{aligned}$$

The first two equations are identical to (3.5) and (3.6), and the last two equations simply state that $d_l(k-\mu_{i,l})$ and $a_l(k-\mu_{i,l})$ should be larger than ε, which they always are. This results in the default order of the train runs: first l, then i.

If $u_{i,l}(k-\mu_{i,l}) = \varepsilon$, then $\overline{u_{i,l}(k-\mu_{i,l})} = 0$ and the equations become:

$$\begin{aligned} d_i(k) &\geq d_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,d},i,l}(k) \otimes \varepsilon = \varepsilon \\ a_i(k) &\geq a_l(k-\mu_{i,l}) \otimes \tau_{\mathrm{h,a},i,l}(k) \otimes \varepsilon = \varepsilon \\ d_l(k-\mu_{i,l}) &\geq d_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \otimes 0 = d_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \\ a_l(k-\mu_{i,l}) &\geq a_i(k) \otimes \tau_{\mathrm{h,d},l,i}(k-\mu_{i,l}) \otimes 0 = a_i(k) \otimes \tau_{\mathrm{h,a},l,i}(k-\mu_{i,l}). \end{aligned}$$

Now the first two equations simply state that $d_i(k)$ and $a_i(k)$ should be larger than ε, which they always are, and the last two equations are equal to equations (3.34) and (3.35). This results in the changed order of train runs: first i, then l.

For two trains running over the same track, and in opposite direction the same procedure of multiplying the right-hand side of the constraints by control inputs can be applied:

$$d_i(k) \geq a_m(k-\mu_{i,m}) \otimes \tau_{\mathrm{s},i,m}(k) \otimes u_{i,m}(k-\mu_{i,m}) \tag{3.40}$$

$$d_m(k-\mu_{i,m}) \geq a_i(k) \otimes \tau_{\mathrm{s},m,i}(k-\mu_{i,m}) \otimes \overline{u_{i,m}(k-\mu_{i,m})}. \tag{3.41}$$

Using this methodology the matrices $A_{\mu,4,\mathrm{d}}$, $A_{\mu,4,\mathrm{a}}$, and $A_{\mu,5}$ can be modified to allow reordering of trains. The new matrices can be described by (3.26) and (3.30) respectively, where $\hat{A}_{\mu,4,\mathrm{d},m} \in \mathbb{R}^{n_{\mathrm{t},m} \times n_{\mathrm{t},m}}$ and $\hat{A}_{\mu,5,m} \in \mathbb{R}^{n_{\mathrm{t},m} \times n_{\mathrm{t},m}}$ are defined as:

$$[\hat{A}_{\mu,4,\mathrm{d},m}]_{i,j} = \begin{cases} \tau_{\mathrm{h,d},T_{m,i},T_{m,j}}(k) \otimes u_{T_{m,i},T_{m,j}}(k-\mu) & \text{if } T_{m,j} \in \mathcal{H}_{T_{m,i}} \text{ and} \\ & \mu = \mu_{T_{m,i},T_{m,j}} \\ \tau_{\mathrm{h,d},T_{m,i},T_{m,j}}(k) \otimes \overline{u_{T_{m,j},T_{m,i}}(k)} & \text{if } T_{m,i} \in \mathcal{H}_{T_{m,j}} \text{ and} \\ & \mu = -\mu_{T_{m,j},T_{m,i}} \\ \varepsilon & \text{else,} \end{cases} \tag{3.42}$$

$$[\hat{A}_{\mu,5,m}]_{i,j} = \begin{cases} \tau_{\mathrm{s},T_{m,i},T_{m,j}}(k) \otimes u_{T_{m,i},T_{m,j}}(k-\mu) & \text{if } T_{m,j} \in \mathcal{S}_{T_{m,i}} \text{ and} \\ & \mu = \mu_{T_{m,i},T_{m,j}} \\ \tau_{\mathrm{s},T_{m,i},T_{m,j}}(k) \otimes \overline{u_{T_{m,j},T_{m,i}}(k)} & \text{if } T_{m,i} \in \mathcal{S}_{T_{m,j}} \text{ and} \\ & \mu = -\mu_{T_{m,j},T_{m,i}} \\ \varepsilon & \text{else,} \end{cases} \tag{3.43}$$

where T_m is again the vector containing the indices of the trains on track m, ordered according to the timetable. The first if-condition in both equations describes the nominal situation where the jth train on track m is in the set $\mathcal{H}_{T_{m,i}}$ or $\mathcal{S}_{T_{m,i}}$ respectively and denotes the headway or separation times between the trains with the added max-plus binary variable $u_{T_{m,i},T_{m,j}}(k-\mu)$. The second if-condition in both equations states that

if the default order of the trains on track m is 'i before j' then element i,j should contain a headway or separation time respectively and an adjoint max-plus binary variable $\overline{u_{T_{m,j},T_{m,i}}(k)}$, since it corresponds to a train order that is different from the nominal order.

For $A_{\mu,4,\mathrm{a}}$, the structure of the matrix is the same as for $A_{\mu,4,\mathrm{d}}$, the only differences are that in (3.42) $\hat{A}_{\mu,4,\mathrm{d},m}$ is replaced by $\hat{A}_{\mu,4,\mathrm{a},m}$, and $\tau_{\mathrm{h,d},T_{m,i},T_{m,j}}(k)$ is replaced by $\tau_{\mathrm{h,a},T_{m,i},T_{m,j}}(k)$.

The reader should note that for a track with n_{train} trains running over it the number of max-plus binary variables added to the model is $n_{\mathrm{train}}(n_{\mathrm{train}}-1)/2$. The number of possible combinations of max-plus binary variables is then $2^{n_{\mathrm{train}}(n_{\mathrm{train}}-1)/2}$, but the number of possible train orders is only $n_{\mathrm{train}}!$. For $n_{\mathrm{train}} \geq 3$ there are more combinations of max-plus binary variables than possible train orders. As a result some combinations of max-plus binary variables do not correspond to possible train orders; they describe infeasible train orders and make the model slightly more complex than absolutely necessary. The reason for adding more max-plus binary variables than necessary is because it is easier to formulate the model: each combination of two trains on a track has one max-plus binary variable that determines the order.

As an example consider again the headway constraints of the three trains running over the same track in the same cycle (so $\mu = 0$ for the headway constraints) with

$$T_1 = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^\top, \quad \mathcal{H}_1 = \emptyset, \quad \mathcal{H}_2 = \{1\}, \quad \mathcal{H}_3 = \{1,2\},$$

and the headway constraints between the departures can be determined with (3.42) resulting in:

$$A_{0,4,\mathrm{d}}(u(k),k) = \begin{bmatrix} \varepsilon & 3 \otimes \overline{u_{2,1}(k)} & 3 \otimes \overline{u_{3,1}(k)} \\ 3 \otimes u_{2,1}(k) & \varepsilon & 3 \otimes \overline{u_{3,2}(k)} \\ 3 \otimes u_{3,1}(k) & 3 \otimes u_{3,2}(k) & \varepsilon \end{bmatrix},$$

where $u_{2,1}(k)$ determines the order between train 1 and 2, $u_{3,2}(k)$ determines the order between train 2 and 3, and $u_{3,1}(k)$ determines the order between train 1 and 3. Now if we set $u_{2,1}(k) = 0$, then train 1 traverses the track before train 2; if we set $u_{3,2}(k) = 0$, then train 2 traverses the track before train 3; and if we set $u_{3,1}(k) = \varepsilon$, then train 3 traverses the track before train 1. But this is impossible because if train 1 traverses the track before train 2 and 2 traverses the track before 3 then $u_{2,1} = 0$ together with $u_{3,2} = 0$ implies that train 1 traverses the track before train 3, which is exactly the opposite of what $u_{3,1} = \varepsilon$ implies. Clearly the combination $u_{2,1} = 0$, $u_{3,2} = 0$, and $u_{3,1} = \varepsilon$ is an infeasible combination.

These infeasible train orders can be derived from the matrix powers of $A_{0,4,\mathrm{d}}$:

$$\begin{aligned} A^{\otimes^2}_{0,4,\mathrm{d}}(u(k),k) &= \begin{bmatrix} 6 \otimes \overline{u_{2,1}(k)} \otimes u_{2,1}(k) & 6 \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} & 6 \otimes \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \\ 6 \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) & 6 \otimes u_{3,2}(k) \otimes \overline{u_{3,2}(k)} & 6 \otimes u_{2,1}(k) \otimes \overline{u_{3,1}(k)} \\ 6 \otimes u_{2,1}(k) \otimes u_{3,2}(k) & 6 \otimes \overline{u_{2,1}(k)} \otimes u_{3,1}(k) & 6 \otimes u_{3,1}(k) \otimes \overline{u_{3,1}(k)} \end{bmatrix} \\ &= \begin{bmatrix} \varepsilon & 6 \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} & 6 \otimes \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \\ 6 \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) & \varepsilon & 6 \otimes u_{2,1}(k) \otimes \overline{u_{3,1}(k)} \\ 6 \otimes u_{2,1}(k) \otimes u_{3,2}(k) & 6 \otimes \overline{u_{2,1}(k)} \otimes u_{3,1}(k) & \varepsilon \end{bmatrix}. \end{aligned}$$

The diagonal elements in $A_{0,4,\mathrm{d}}^{\otimes^2}(u(k),k)$ are ε by definition since $\overline{u_i(k)} \otimes u_i(k) = \varepsilon$ and $\varepsilon \otimes a = \varepsilon$. The next matrix power of $A_{0,4,\mathrm{d}}$ is:

$$A_{0,4,\mathrm{d}}^{\otimes^3}(u(k),k) = \begin{bmatrix} AP_1(u(k),k) & AP_2(u(k),k) & AP_3(u(k),k) \end{bmatrix},$$

with

$$AP_1(u(k),k) = \begin{bmatrix} 9 \otimes (u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} \oplus \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k)) \\ 9 \otimes u_{2,1}(k) \otimes \underline{u_{3,2}(k)} \otimes \overline{u_{3,2}(k)} \\ 9 \otimes u_{3,2}(k) \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) \end{bmatrix}$$

$$AP_2(u(k),k) = \begin{bmatrix} 9 \otimes \overline{u_{2,1}(k)} \otimes u_{3,1}(k) \otimes \overline{u_{3,1}(k)} \\ 9 \otimes (u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} \oplus \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k)) \\ 9 \otimes u_{3,2}(k) \otimes u_{3,1}(k) \otimes \overline{u_{3,1}(k)} \end{bmatrix}$$

$$AP_3(u(k),k) = \begin{bmatrix} 9 \otimes u_{2,1}(k) \otimes \overline{u_{2,1}(k)} \otimes \overline{u_{3,1}(k)} \\ 9 \otimes u_{2,1}(k) \otimes \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \\ 9 \otimes (u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} \oplus \overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k)) \end{bmatrix}.$$

All non-diagonal elements of $A_{0,4,\mathrm{d}}^{\otimes^3}(u(k),k)$ are ε by definition. The diagonal elements are positive if $u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} = 0$ or $\overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) = 0$. That means these combinations of max-plus binary variables correspond to infeasible train orders. So by simply determining the matrix powers, which has to be done to determine the explicit model anyway, and looking at the diagonal elements the infeasible combinations of max-plus binary variables can be found.

3.3.2 Breaking connections

Breaking connections can be done by manipulating the entries of $A_{\mu,3}$. By setting the elements of $A_{\mu,3}$ to ε the connection is broken. This can be done by adding a max-plus binary variable $u_{i,j+n}(k-\mu_{i,j}) \in \{\varepsilon, 0\}$ to the connection constraint, as was already shown by de Vries et al. [27] and Heidergott and Vries [44]. If $u_{i,j+n}(k-\mu_{i,j+n}) = \varepsilon$ the connection is broken, if $u_{i,j+n}(k-\mu_{i,j+n}) = 0$ the connection is maintained. This results in a new matrix:

$$[A_{\mu,3}]_{i,j} = \begin{cases} \tau_{\mathrm{c},i,j}(k) \otimes u_{i,j+n}(k-\mu_{i,j}) & \text{if } j \in \mathcal{C}_i \text{ and } \mu = \mu_{i,j} \\ \varepsilon & \text{else.} \end{cases} \tag{3.44}$$

The max-plus binary variable has indices $(i, j+n)$ since element $[A_{\mu,3}]_{i,j}$ has indices $(i, j+n)$ in A_μ. This ensures there is no overlap in the indices of the max-plus binary variables for the different dispatching actions.

3.3.3 Coupling trains

At some stations two 'physical' trains are coupled and continue as a single 'physical' train; this is modeled by coupling constraints. When one of the two trains is delayed it may be better not to couple the trains. To ensure the trains are not coupled the coupling constraints need to be removed and headway constraints between the two trains need to

be added. These headway constraints should also allow the trains to depart in a different order. For the decoupling new max-plus binary variables are used: $v_{i,j}(k) \in \{\varepsilon, 0\}$ and its adjoint $\overline{v_{i,j}(k)} \in \{\varepsilon, 0\}$. These max-plus binary variables are defined as in (2.18). This results in the following build-up of the coupling matrix $A_{\mu,6,d}$:

$$[A_{\mu,6,\mathrm{d}}]_{i,j} = \begin{cases} 0 \otimes u_{i,j}(k) \oplus \tau_{\mathrm{h,d},i,j}(k) \otimes \overline{v_{i,j}(k)} \otimes u_{i,j}(k) & \text{if } j = o_i,\ i > j, \\ & \text{and } \mu = 0 \\ 0 \otimes u_{i,j}(k) \oplus \tau_{\mathrm{h,d},i,j}(k) \otimes \overline{v_{i,j}(k)} \otimes \overline{u_{i,j}(k)} & \text{if } i = o_j,\ i < j, \\ & \text{and } \mu = 0 \\ \varepsilon & \text{else,} \end{cases} \tag{3.45}$$

where $v_{i,j}(k)$ is the max-plus binary variable for coupling and $v_{i,j}(k)$ is the max-plus binary variable that determines the order of the train departures if the trains are not coupled. The matrices $\hat{A}_{\mu,6,\mathrm{a}}$ are defined in the same way as $\hat{A}_{\mu,6,\mathrm{d}}$ with only one difference: $\tau_{\mathrm{h,d},i,j}(k)$ is replaced by $\tau_{\mathrm{h,a},i,j}(k)$.

Clearly if $v_{i,j}(k) = 0$ then $\overline{v_{i,j}(k)} = \varepsilon$ and the trains remain coupled. If $\overline{v_{i,j}(k)} = 0$ then either $[A_{\mu,6}, \mathrm{d}]_{i,j} = \tau_{\mathrm{h,d},i,j}(k)$ and $[A_{\mu,6}]_{j,i} = \varepsilon$ or $[A_{\mu,6}]_{i,j} = \varepsilon$ and $[A_{\mu,6,\mathrm{d}}]_{j,i} = \tau_{\mathrm{h,d},j,i}(k)$.

3.3.4 Switching between tracks

Between certain stations in a railway network, there may be two (or more) parallel tracks that can be used by trains. To determine which track should be used, extra max-plus binary variables are added per train. In this thesis we will be dealing with at most two parallel tracks in each direction (two sets of two unidirectional tracks). This is done for the sake of simplicity. As a result only one max-plus binary variable has to be added per train traversing one of these parallel tracks. However, this approach can easily be generalized to the case where there are more parallel tracks. In that case one max-plus binary variable per track per train is added that indicates whether the train is on that track. Although this will result in many more max-plus binary variables than needed in theory, it will keep the constraints as simple as possible.

The tracks are numbered in such a way that parallel tracks always have consecutive numbers. If trains can switch between track m and $m+1$, then this can be modeled by changing the headway matrices $A_{\mu,4,\mathrm{d}}$. Consider the part of $A_{\mu,4,\mathrm{d}}$, as described in (3.26), consisting of

$$\tilde{A}_{\mu,4,\mathrm{d},m} = \begin{bmatrix} \hat{A}_{\mu,4,\mathrm{d},m} & \mathcal{E} \\ \mathcal{E} & \hat{A}_{\mu,4,\mathrm{d},m+1} \end{bmatrix},$$

where $\hat{A}_{\mu,4,\mathrm{d},m}$ and $\hat{A}_{\mu,4,\mathrm{d},m+1}$ contain the headway times for the departures of the trains on track m and $m+1$, respectively. Define T_m as the vector containing the indices of the trains on track m, ordered according to the timetable and $\tilde{T}_m = [T_m^\top T_{m+1}^\top]^\top$ is the vector containing the indices of the trains on track m and track $m+1$. To enable switching between parallel tracks new max-plus binary variables are introduced: $w_{i,j}(k)$ and its adjoint $\overline{w_{i,j}(k)}$. These max-plus binary variables are defined as in (2.18). Switching

between tracks can then be done by redefining the entries of $\tilde{A}_{\mu,4,\mathrm{d},m}$ as follows:

$$\left[\tilde{A}_{\mu,4,\mathrm{d},m}\right]_{i,j} = \begin{cases} \tau_{\mathrm{h,d},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)\otimes & \\ u_{\tilde{T}_{m,i},\tilde{T}_{m,j}}(k-\mu)\otimes & \text{if } \tilde{T}_{m,j}\in\mathcal{H}_{\tilde{T}_{m,i}} \text{ and} \\ \Big(w_{\tilde{T}_{m,i}}(k)\otimes w_{\tilde{T}_{m,j}}(k-\mu)\oplus & \mu=\mu_{\tilde{T}_{m,i},\tilde{T}_{m,j}} \\ \overline{w_{\tilde{T}_{m,i}}(k)}\otimes\overline{w_{\tilde{T}_{m,j}}(k-\mu)}\Big) & \\ \tau_{\mathrm{h,d},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)\otimes & \\ \overline{u_{\tilde{T}_{m,j},\tilde{T}_{m,i}}(k)}\otimes & \text{if } \tilde{T}_{m,i}\in\mathcal{H}_{\tilde{T}_{m,j}} \text{ and} \\ \Big(w_{\tilde{T}_{m,i}}(k)\otimes w_{\tilde{T}_{m,j}}(k-\mu)\oplus & \mu=-\mu_{\tilde{T}_{m,j},\tilde{T}_{m,i}} \\ \overline{w_{\tilde{T}_{m,i}}(k)}\otimes\overline{w_{\tilde{T}_{m,j}}(k-\mu)}\Big) & \\ \tau_{\mathrm{h,d},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)\otimes & \text{if } r_{\mathrm{d},\tilde{T}_{m,i}}(k)\geq r_{\mathrm{d},\tilde{T}_{m,j}}(k-\mu), \\ u_{\tilde{T}_{m,i},\tilde{T}_{m,j}}(k-\mu)\otimes & \mu=\mu_{\tilde{T}_{m,i},\tilde{T}_{m,j}}\geq 0 \text{ and} \\ \Big(w_{\tilde{T}_{m,i}}(k)\otimes\overline{w_{\tilde{T}_{m,j}}(k-\mu)}\oplus & \tilde{T}_{m,i}\in\mathcal{H}_{T_m},\ \tilde{T}_{m,j}\in\mathcal{H}_{T_{m+1}} \text{ or} \\ \overline{w_{\tilde{T}_{m,i}}(k)}\otimes w_{\tilde{T}_{m,j}}(k-\mu)\Big) & \tilde{T}_{m,i}\in\mathcal{H}_{T_{m+1}},\ \tilde{T}_{m,j}\in\mathcal{H}_{T_m} \\ \tau_{\mathrm{h,d},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)\otimes & \text{if } r_{\mathrm{d},\tilde{T}_{m,j}}(k-\mu)>r_{\mathrm{d},\tilde{T}_{m,i}}(k), \\ \overline{u_{\tilde{T}_{m,j},\tilde{T}_{m,i}}(k)}\otimes & \mu=-\mu_{\tilde{T}_{m,j},\tilde{T}_{m,i}}\leq 0 \text{ and} \\ \Big(w_{\tilde{T}_{m,i}}(k)\otimes\overline{w_{\tilde{T}_{m,j}}(k-\mu)}\oplus & \tilde{T}_{m,i}\in\mathcal{T}_m,\ \tilde{T}_{m,j}\in\mathcal{H}_{T_{m+1}} \text{ or} \\ \overline{w_{\tilde{T}_{m,i}}(k)}\otimes w_{\tilde{T}_{m,j}}(k-\mu)\Big) & \tilde{T}_{m,i}\in\mathcal{H}_{T_{m+1}},\ \tilde{T}_{m,j}\in\mathcal{H}_{T_m}. \end{cases} \tag{3.46}$$

The max-plus binary variables $u_{\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)$, $u_{\tilde{T}_{m,i},\tilde{T}_{m,j}}(k-\mu)$ are used to determine the order of the trains $\tilde{T}_{m,i}$ and $\tilde{T}_{m,j}$. The max-plus binary variables $w_{\tilde{T}_{m,i}}(k)$ and $w_{\tilde{T}_{m,j}}(k-\mu)$ are used to determine on which track trains $\tilde{T}_{m,i}$ and $\tilde{T}_{m,j}$ are respectively. The values are chosen such that if $w_{\tilde{T}_{m,j}}(k)=0$ and $w_{\tilde{T}_{m,j}}(k-\mu)=0$, trains $\tilde{T}_{m,i}$ and $\tilde{T}_{m,j}$ are on the same track as in the nominal case. The first two if-statements describe the headway constraints for trains that are on the same track during nominal operation with the added max-plus binary variables for the different tracks. The last two if-statements are the headway constraints between trains that, during the nominal operation, traverse parallel tracks. The default order between these trains is chosen according to their scheduled departure times.

For $\tilde{A}_{\mu,4,\mathrm{a},m}$, the structure of the matrix is the same as for $\tilde{A}_{\mu,4,\mathrm{d},m}$, the only difference is that in (3.46) $\tau_{\mathrm{h,d},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)$ is replaced by $\tau_{\mathrm{h,a},\tilde{T}_{m,i},\tilde{T}_{m,j}}(k)$.

To illustrate the method of adding and removing headway constraints based on the track the trains are on consider the following example consisting of two parallel tracks, track 1 and 2, with two trains on each of the tracks. Trains 1 and 2 traverse track 1 and trains 3 and 4 traverse track 2. All trains are running in the same cycle. The timetable period is 1 hour, and the timetable vector is $r(0)=\begin{bmatrix}0 & 30 & 5 & 35 & 15 & 50 & 20 & 55\end{bmatrix}^\top$. We

can define the following sets and vectors based on this example:

$$T_1 = \begin{bmatrix} 1 & 2 \end{bmatrix}^\top, \qquad T_2 = \begin{bmatrix} 3 & 4 \end{bmatrix}^\top, \qquad \tilde{T}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}^\top,$$
$$\mathcal{H}_1 = \emptyset, \qquad \mathcal{H}_2 = \{1\}, \qquad \mathcal{H}_3 = \emptyset, \qquad \mathcal{H}_4 = \{3\}.$$

Using these sets, vectors and (3.46) we can determine the entries of $\tilde{A}_{\mu,4,\mathrm{d},1}$. Since all trains are in the same cycle only $\mu = 0$ needs to be considered. The first if-condition of (3.46) results in the following non-ε elements of $\tilde{A}_{0,4,\mathrm{d},1}$:

$$[\tilde{A}_{0,4,\mathrm{d},1}]_{1,2} = \tau_{\mathrm{h,d},1,2}(k) \otimes \overline{u_{2,1}(k)} \otimes \Big(w_1(k) \otimes w_2(k) \oplus \overline{w_1(k)} \otimes \overline{w_2(k)}\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{3,4} = \tau_{\mathrm{h,d},3,4}(k) \otimes \overline{u_{4,3}(k)} \otimes \Big(w_3(k) \otimes w_4(k) \oplus \overline{w_3(k)} \otimes \overline{w_4(k)}\Big),$$

and the second if-condition results in the following non-ε elements of $\tilde{A}_{0,4,\mathrm{d},1}$:

$$[\tilde{A}_{0,4,\mathrm{d},1}]_{2,1} = \tau_{\mathrm{h,d},2,1}(k) \otimes u_{2,1}(k) \otimes \Big(w_1(k) \otimes w_2(k) \oplus \overline{w_1(k)} \otimes \overline{w_2(k)}\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{4,3} = \tau_{\mathrm{h,d},4,3}(k) \otimes u_{4,3}(k) \otimes \Big(w_3(k) \otimes w_4(k) \oplus \overline{w_3(k)} \otimes \overline{w_4(k)}\Big).$$

The first two if-conditions result in the headway times between departure events of the trains that traverse the same track during nominal operation, with the addition of the max-plus binary variables for track selection. The third if-condition results in the following non-ε elements $\tilde{A}_{0,4,\mathrm{d},1}$:

$$[\tilde{A}_{0,4,\mathrm{d},1}]_{1,3} = \tau_{\mathrm{h,d},1,3}(k) \otimes \overline{u_{3,1}(k)} \otimes \Big(w_1(k) \otimes \overline{w_3(k)} \oplus \overline{w_1(k)} \otimes w_3(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{1,4} = \tau_{\mathrm{h,d},1,4}(k) \otimes \overline{u_{4,1}(k)} \otimes \Big(w_1(k) \otimes \overline{w_4(k)} \oplus \overline{w_1(k)} \otimes w_4(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{2,4} = \tau_{\mathrm{h,d},2,4}(k) \otimes \overline{u_{4,2}(k)} \otimes \Big(w_2(k) \otimes \overline{w_4(k)} \oplus \overline{w_2(k)} \otimes w_4(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{3,2} = \tau_{\mathrm{h,d},3,2}(k) \otimes \overline{u_{2,3}(k)} \otimes \Big(w_3(k) \otimes \overline{w_2(k)} \oplus \overline{w_3(k)} \otimes w_2(k)\Big),$$

and the fourth if-condition results in the final non-ε elements $\tilde{A}_{0,4,\mathrm{d},1}$:

$$[\tilde{A}_{0,4,\mathrm{d},1}]_{3,1} = \tau_{\mathrm{h,d},3,1}(k) \otimes u_{3,1}(k) \otimes \Big(w_1(k) \otimes \overline{w_3(k)} \oplus \overline{w_1(k)} \otimes w_3(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{4,1} = \tau_{\mathrm{h,d},4,1}(k) \otimes u_{4,1}(k) \otimes \Big(w_1(k) \otimes \overline{w_4(k)} \oplus \overline{w_1(k)} \otimes w_4(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{4,2} = \tau_{\mathrm{h,d},4,2}(k) \otimes u_{4,2}(k) \otimes \Big(w_2(k) \otimes \overline{w_4(k)} \oplus \overline{w_2(k)} \otimes w_4(k)\Big)$$
$$[\tilde{A}_{0,4,\mathrm{d},1}]_{2,3} = \tau_{\mathrm{h,d},2,3}(k) \otimes u_{2,3}(k) \otimes \Big(w_3(k) \otimes \overline{w_2(k)} \oplus \overline{w_3(k)} \otimes w_2(k)\Big).$$

These headway times define the order of the trains if a train from track 1 switches to track 2 or the other way around. For the headway times between arrival events only $\tau_{\mathrm{h,d},i,j}(k)$ needs to be replaced with $\tau_{\mathrm{h,a},i,j}(k)$.

When trains 1 and 2 remain on track 1, and trains 3 and 4 remain on track 2 the max-plus binary variables $w_1(k)$, $w_2(k)$,$w_3(k)$, and $w_4(k)$ are all zero (and $\overline{w_1(k)}$, $\overline{w_1(k)}$,

$\overline{w_1(k)}$, and $\overline{w_1(k)}$ are ε), resulting in the following headway constraints:

$$\begin{aligned}
[\tilde{A}_{0,4,\mathrm{d},1}]_{1,2} &= \tau_{\mathrm{h,d},1,2}(k) \otimes \overline{u_{2,1}(k)} & [\tilde{A}_{0,4,\mathrm{d},1}]_{2,4} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{3,4} &= \tau_{\mathrm{h,d},3,4}(k) \otimes \overline{u_{4,3}(k)} & [\tilde{A}_{0,4,\mathrm{d},1}]_{3,2} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{2,1} &= \tau_{\mathrm{h,d},2,1}(k) \otimes u_{2,1}(k) & [\tilde{A}_{0,4,\mathrm{d},1}]_{3,1} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{4,3} &= \tau_{\mathrm{h,d},4,3}(k) \otimes u_{4,3}(k) & [\tilde{A}_{0,4,\mathrm{d},1}]_{4,1} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{1,3} &= \varepsilon & [\tilde{A}_{0,4,\mathrm{d},1}]_{4,2} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{1,4} &= \varepsilon & [\tilde{A}_{0,4,\mathrm{d},1}]_{2,3} &= \varepsilon.
\end{aligned}$$

Leaving only the headway constraints between trains 1 and 2, and the headway constraints between trains 3 and 4.

Now if train 3 changes tracks to track 1, then the max-plus binary variable $w_3(k)$ changes to ε (and $\overline{w_3(k)} = 0$) and the headway constraints become:

$$\begin{aligned}
[\tilde{A}_{0,4,\mathrm{d},1}]_{1,2} &= \tau_{\mathrm{h,d},1,2}(k) \otimes \overline{u_{2,1}(k)} & [\tilde{A}_{0,4,\mathrm{d},1}]_{3,4} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{2,1} &= \tau_{\mathrm{h,d},2,1}(k) \otimes u_{2,1}(k) & [\tilde{A}_{0,4,\mathrm{d},1}]_{4,3} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{1,3} &= \tau_{\mathrm{h,d},1,3}(k) \otimes \overline{u_{3,1}(k)} & [\tilde{A}_{0,4,\mathrm{d},1}]_{1,4} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{3,2} &= \tau_{\mathrm{h,d},3,2}(k) \otimes \overline{u_{2,3}(k)} & [\tilde{A}_{0,4,\mathrm{d},1}]_{2,4} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{3,1} &= \tau_{\mathrm{h,d},3,1}(k) \otimes u_{3,1}(k) & [\tilde{A}_{0,4,\mathrm{d},1}]_{4,1} &= \varepsilon \\
[\tilde{A}_{0,4,\mathrm{d},1}]_{2,3} &= \tau_{\mathrm{h,d},2,3}(k) \otimes u_{3,2}(k) & [\tilde{A}_{0,4,\mathrm{d},1}]_{4,2} &= \varepsilon.
\end{aligned}$$

Clearly the headway constraints between train 3 and 4 have been turned 'off' and headway constraints between train 3 and trains 1 and 2 have been turned 'on' such that the default order of departure on track 1 is: train 1 first, train 3 second, and train 2 third.

3.4 Explicit switching max-plus-linear model

The model introduced in (3.33) has a specific structure called the implicit form. In an equation in the implicit form the state vector $x(k)$ does not only depend on the state vector of the previous cycles (and the timetable reference), but also on itself. In this section we will first describe how the implicit SMPL for a single cycle can be converted into its explicit form and after that the method is extended to an implicit SMPL for multiple cycles.

3.4.1 Explicit model for a single cycle

When only the event times in $x(k)$ need to be determined only a single cycle of the (switching) max-plus-linear model needs to be considered. For a model of a single cycle only the state vectors of the current and past cycles needs to be considered and no state vectors of future cycles, i.e. only $x(k-\mu)$ with $\mu < 0$, are considered. As a result, A_μ is only needed for $\mu = 0, \ldots, \mu_{\max}$. This reduces the implicit model to

$$x(k) = r(k) \oplus A_0 \otimes x(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k-\mu).$$

Next, A_0^* will need to be determined. In Section 2.2 it was shown how A^* can be determined for a constant matrix by using Theorem 3.17 and Theoremm 3.20 of Baccelli et al. [3]. For the given model calculating the max-plus matrix powers $A_0^{\otimes p}$ can be done in the same way as for a constant matrix. In order to calculate max-plus matrix powers the possibility of infinite event times needs to be considered, since infinite event times may be caused by circuits of positive weight in the graph of A_0 and a requirement for calculating A_0^* is that circuits of positive weight do not exist in the graph of A_0. If one or more event times are $+\infty$, the timetable is infeasible. Since the nominal model results in a feasible timetable, infinite event times can only result from the max-plus binary variables. More specifically, infinite event times can only have two causes: infinite process times or positive diagonal elements in one of the max-plus matrix powers $A_0^{\otimes p}$ (positive diagonal elements in the matrix powers of A_0 correspond to circuits of positive weight). Since none of the process times in the model can be infinite, the only cause that remains is positive diagonal elements. It is therefore clear that the combinations of max-plus binary variables resulting in infinite event times also result in positive diagonal elements of $A_0^{\otimes p}$ for $p \in \{1, \ldots, \infty\}$. Hence, it is necessary to determine the value combinations of max-plus binary variables that result in positive diagonal elements when calculating A_0^* and remove all elements containing these specific value combinations of max-plus binary variables from the matrix powers of A_0. This can be done by simply calculating the matrix powers of A_0 and each time a non-ε diagonal element is in the matrix, check which value combination(s) of max-plus binary variables the diagonal elements consist of and set all elements of the matrix powers of A that have those value combinations of max-plus binary variables to ε.

By removing these infeasible value combinations of inputs from the model a feasible explicit switching max-plus-linear model of the following form is found:

$$x(k) = A_0^{*,\text{feas}} \otimes \left(r(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k-\mu) \right), \tag{3.47}$$

where $A_0^{*,\text{feas}}$ is the feasible part of A_0^* (so the infeasible combinations of max-plus binary variables are removed from the matrix).

Consider the example of the three trains traversing the same track given in Section 3.3.1. The following value combinations of max-plus binary variables correspond to infeasible train orders:

$$u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} = 0$$
$$\overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) = 0.$$

These value combinations of max-plus binary variables can be removed by replacing $u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)}$ and $\overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k)$ with ε.

To ensure the model predictive controller does not use these value combinations of max-plus binary variables we add constraints to the model predictive controller that ensure the following max-plus-linear inequalities are satisfied:

$$\overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) \leq \varepsilon$$
$$u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} \leq \varepsilon.$$

3.4.2 Explicit model for multiple cycles

Some dispatching actions may have an effect on the event times in the next cycle, or even the cycles after that, and therefore it may be needed to consider multiple cycles of the switching max-plus-linear model. Multiple cycles can be considered at once by extending the model which will be done in this part of the chapter.

A model with multiple cycles is used to predict the arrival and departure times for the current and future cycles and the effects of the dispatching actions on these arrival and departure times based on the current situation. That means $x(k)$ and the event times of subsequent cycles ($x(k-\mu)$, with $\mu < 0$) need to be determined. The model for $m+1$ cycles can be described by the following set of equations:

$$x(k+q) = r(k+q) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k+q-\mu) \oplus \bigoplus_{\mu=-\min(\mu_{\max},m-q)}^{0} A_\mu \otimes x(k+q-\mu), \tag{3.48}$$

for the range $q = 0,\ldots,m$. By extending the state vector to

$$\breve{x}(k) = \begin{bmatrix} x^\top(k) & x^\top(k+1) & \ldots & x^\top(k+m-1) & x^\top(k+m) \end{bmatrix}^\top,$$

the above set of equations can be written as

$$\breve{x}(k) = \breve{r}(k) \oplus \breve{A}_0 \otimes \breve{x}(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \breve{A}_\mu \otimes x(k-\mu), \tag{3.49}$$

where $\breve{A}_0(k, v(k))$ contains the matrices with the process times for the constraints between event times of the current and future cycles ($x(k-\mu)$, with $\mu \leq 0$), and the max-plus sum with μ ranging from 1 to $\mu_{\max}$ contains the matrices with the process times for the constraints between event times of preceding cycles ($x(k-\mu)$, with $\mu > 0$) and event times of the current and future cycles. These matrices are defined as follows:

$$[\breve{A}_0(k,u(k))]_{i,j} = \begin{cases} A_{i-j}(k+i-1, u(k+j-1)) & \text{if } 0 \leq i-j \leq \mu_{\max} \\ & \text{and } 1 \leq i,j \leq m+1 \\ A_{i-j}(k+i-1, u(k+i-1)) & \text{if } -\mu_{\max} \leq i-j \leq -1 \\ & \text{and } 1 \leq i,j \leq m+1 \\ \mathcal{E} & \text{else,} \end{cases}$$

and

$$\breve{A}_\mu = \begin{bmatrix} A_\mu^\top & A_{\mu+1}^\top & \ldots & A_{\mu_{\max}}^\top & \mathcal{E}^\top & \ldots & \mathcal{E}^\top \end{bmatrix}^\top.$$

The reference vector $\breve{r}(k)$ containing the planned event times is defined as follows:

$$\breve{r}(k) = \begin{bmatrix} r^\top(k) & r^\top(k+1) & \ldots & r^\top(k+m-1) & r^\top(k+m) \end{bmatrix}^\top.$$

The same procedure as for the model of one cycle can now be applied to $\breve{A}_0$, resulting in $\breve{A}_0^{*,\text{feas}}$ and a feasible explicit switching max-plus-linear model for multiple cycles:

$$\breve{x}(k) = \breve{A}_0^{*,\text{feas}} \otimes \left(\breve{r}(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \breve{A}_\mu \otimes x(k-\mu) \right). \tag{3.50}$$

For example if we want to determine the event times in the current cycle k and the next cycles $k+1$ and $k+2$ then according to (3.48) with $m=2$ the following set of equations needs to be solved:

$$\begin{aligned}
x(k) &= r(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k-\mu) \oplus \bigoplus_{\mu=-2}^{0} A_\mu \otimes x(k-\mu) \\
&= r(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k-\mu) \oplus A_{-2} \otimes x(k+2) \oplus A_{-1} \otimes x(k+1) \oplus A_0 \otimes x(k) \\
x(k+1) &= r(k+1) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k+1-\mu) \oplus \bigoplus_{\mu=-1}^{0} A_\mu \otimes x(k+1-\mu) \\
&= r(k+1) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k+1-\mu) \oplus A_{-1} \otimes x(k+2) \oplus A_0 \otimes x(k+1) \\
x(k+2) &= r(k+2) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k+2-\mu) \oplus \bigoplus_{\mu=0}^{0} A_\mu \otimes x(k+2-\mu) \\
&= r(k+2) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} A_\mu \otimes x(k+2-\mu) \oplus A_0 \otimes x(k+2).
\end{aligned}$$

Now by defining the extended state vector as:

$$\breve{x}(k) = \begin{bmatrix} x^\top(k) & x^\top(k+1) & x^\top(k+2) \end{bmatrix}^\top,$$

and the extended reference vector as:

$$\breve{r}(k) = \begin{bmatrix} r^\top(k) & r^\top(k+1) & r^\top(k+2) \end{bmatrix}^\top.$$

the extended switching max-plus-linear model can be described by (3.49) with system matrices:

$$\breve{A}_0(k,u(k)) = \begin{bmatrix} A_0(k,u(k)) & A_{-1}(k,u(k)) & A_{-2}(k,u(k)) \\ A_1(k+1,u(k)) & A_0(k+1,u(k+1)) & A_{-1}(k+1,u(k+1)) \\ A_2(k+2,u(k)) & A_1(k+2,u(k+1)) & A_0(k+2,u(k+2)) \end{bmatrix}.$$

and

$$\breve{A}_\mu = \begin{bmatrix} A_\mu^\top & A_{\mu+1}^\top & \dots & A_{\mu_{\max}}^\top & \mathcal{E}^\top & \dots & \mathcal{E}^\top \end{bmatrix}^\top.$$

3.4.3 Structured approach to matrix multiplication

The calculation of the matrix powers of A_0 can be done in a structured manner by making use of graph theory and the structure of the matrix as shown in (3.20) and the sub matrices that are described in Sections 3.2 and 3.3. First denote $A_{0,4,\mathrm{d}} \oplus A_{0,6,\mathrm{d}}$ as A_a, $A_{0,2} \oplus A_{0,3} \oplus A_{0,5}$ as A_b, $A_{0,1}$ as A_c, and $A_{0,4,\mathrm{a}} \oplus A_{0,6,\mathrm{a}}$ as A_d.

Then matrix A_0 can be written as:

$$A_0 = \begin{bmatrix} A_\mathrm{a} & A_\mathrm{b} \\ A_\mathrm{c} & A_\mathrm{d} \end{bmatrix},$$

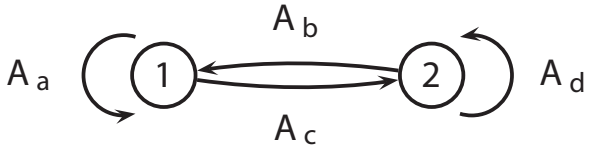

Figure 3.1: Graph $\mathcal{G}(A_0)$.

The graph $\mathcal{G}(A_0)$ is given in Figure 3.1, where each sub-matrix A_a, A_b, A_c, and A_d is considered as a single edge of length 1 between the nodes. According to graph theory $[A^{\otimes m}]_{i,j}$ corresponds to the maximum weight of all paths of length m from node j to node i in the precedence graph of A. If we apply this to matrix A_0 for element $[A_0^{\otimes 2}]_{1,1}$, then we should look at all paths of length 2 in the graph that start and end at node 1. There are two paths of length 2 from node 1 to node 1. One consists of taking the edge from node 1 to node 1 twice resulting in a weight of $A_\mathrm{a}^{\otimes 2}$. The other path of length two consists of taking the edge from node 1 to node 2 and the edge from node 2 to node, resulting in a path weight of $A_\mathrm{b} \otimes A_\mathrm{c}$. The value of $[A_0^{\otimes 2}]_{1,1}$ is the maximum of the weight of both paths: $[A_0^{\otimes 2}]_{1,1} = A_\mathrm{a}^{\otimes 2} \oplus A_\mathrm{b} \otimes A_\mathrm{c}$. For the other elements of $A_\mathrm{a}^{\otimes 2}$ this results in:

$$
\begin{aligned}
[A_0^{\otimes 2}]_{1,2} &= A_\mathrm{a} \otimes A_\mathrm{b} \oplus A_\mathrm{b} \otimes A_\mathrm{d} \\
[A_0^{\otimes 2}]_{2,1} &= A_\mathrm{c} \otimes A_\mathrm{a} \oplus A_\mathrm{d} \otimes A_\mathrm{c} \\
[A_0^{\otimes 2}]_{2,2} &= A_\mathrm{d}^{\otimes 2} \oplus A_\mathrm{c} \otimes A_\mathrm{b}.
\end{aligned}
$$

For the elements of $A_0^{\otimes 3}$ we should look at all paths of length 3. There are four path of length 3 starting at node 1 and ending at node 1:

- Take the edge from node 1 to node 1 three times. This path has a weight of $A_\mathrm{a}^{\otimes 3}$.
- Take the edge from node 1 to node 1 a single time and then go to node 2 and back. This path has a weight of $A_\mathrm{a} \otimes A_\mathrm{b} \otimes A_\mathrm{c}$.
- First go to node 2 and back, then take the edge from 1 node to node 1. This path has a weight of $A_\mathrm{b} \otimes A_\mathrm{c} \otimes A_\mathrm{a}$.
- Take the edge from node 1 to node 2, then the edge from node 2 to node 2, and finally the edge from node 2 to node 1. This path has a weight of $A_\mathrm{b} \otimes A_\mathrm{d} \otimes A_\mathrm{c}$.

This can also be done for the other elements and for all elements of all other matrix powers. By looking at the graph of A_0 and the relation between paths in the graph and elements of the matrix powers of the matrix, it is clear only a limited number of paths is possible and that there is a clear structure in these paths.

This structure follows a given set of rules that can directly be derived from graph theory.

For paths starting and ending in node 1 the following rules apply:

1) The number of times edge A_c from node 1 to node 2 is in the path must be equal to the number of times edge A_b from node 2 to node 1 is in the path.

2) If edge A_{d} from node 2 to node 2 is in the path then edge A_{b} and edge A_{c} must also be in the path.

3) Edge A_{a} from node 1 to node 1 can only be in the path after edge A_{c}, and after itself.

4) Edge A_{d} from node 2 to node 2 can only be in the path after edge A_{b}, and after itself.

5) Edge A_{c} from node 1 to node 2 can only be at the start of the path or after edge A_{a} or edge A_{b}.

6) Edge A_{b} from node 2 to node 1 can only be in the path after edge A_{d} or edge A_{c}.

7) The path must start with edge A_{a} or edge A_{c}.

8) The path must end with edge A_{a} or edge A_{b}.

9) The number of edges is equal to the matrix power.

This translates into the following equations for $[A_0^{\otimes m}]_{1,1}$:

$$[A_0^{\otimes m}]_{1,1} = \bigoplus_{(q_1,q_2,q_3)\in\mathcal{S}_{1,1,m}} \bigotimes_{i=1}^{l} A_{\mathrm{a}}^{\otimes q_{1,i}} \otimes A_{\mathrm{b}}^{\otimes q_{2,i}} \otimes A_{\mathrm{d}}^{\otimes q_{3,i}} \otimes A_{\mathrm{c}}^{\otimes q_{2,i}}, \tag{3.51}$$

where $q_{1,i} = (q_1)_i$, $q_{2,i} = (q_2)_i$, $q_{3,i} = (q_3)_i$, and where the set $\mathcal{S}_{1,1,m}$ contains all tuples (q_1, q_2, q_3) of vectors that satisfy the following equation:

$$\sum_{i=1}^{l} q_{1,i} + 2q_{2,i} + q_{3,i} = m, \tag{3.52}$$

with $q_{1,i} \in \{0,\ldots,m\}$, $q_{2,i} \in \{0,1\}$, and $q_{3,i} = 0$ if $q_{2,i} = 0$ and $q_{3,i} \in \{0,\ldots,m\}$ if $q_{2,i} = 1$.

Rule 1 is ensured by setting the max-plus power of A_{b} equal to that of A_{c}. Rule 2 ensured by the if-condition on $q_{3,i}$. Rules 3-8 are ensured by the order in which A_{a}, A_{b}, A_{c}, and A_{d} are put in 3.51. The sum in (3.52) ensures rule 9.

The rules for paths starting and ending in node 2 are:

1) The number of times edge A_{c} from node 1 to node 2 is in the path must be equal to the number of times edge A_{b} from node 2 to node 1 is in the path.

2) If edge A_{a} from node 1 to node 1 is in the path then edge A_{b} from node 1 to node 2 and edge A_{c} from node 2 to node 1 must also be in the path.

3) Edge A_{a} from node 1 to node 1 can only be in the path after edge A_{c}, and after itself.

4) Edge A_{d} from node 2 to node 2 can only be in the path after edge A_{b}, and after itself.

5) Edge A_{c} from node 1 to node 2 can only be after edge A_{a} or edge A_{b}.

6) Edge A_b from node 2 to node 1 can only be at the start the path or after edge A_d or edge A_c.

7) The path must start with edge A_d or edge A_b.

8) The path must end with edge A_d or edge A_c.

9) The number of edges is equal to the matrix power.

These rules result in:

$$[A_0^{\otimes m}]_{2,2} = \bigoplus_{(q_1,q_2,q_3)\in\mathcal{S}_{2,2,m}} \bigotimes_{i=1}^{l} A_\mathrm{d}^{\otimes q_{1,i}} \otimes A_\mathrm{c}^{\otimes q_{2,i}} \otimes A_\mathrm{a}^{\otimes q_{3,i}} \otimes A_\mathrm{b}^{\otimes q_{2,i}}, \tag{3.53}$$

where $q_{1,i} = (q_1)_i$, $q_{2,i} = (q_2)_i$, $q_{3,i} = (q_3)_i$, and where the set $\mathcal{S}_{2,2,m}$ contains all tuples (q_1, q_2, q_3) that satisfy the following equation:

$$\sum_i^l q_{1,i} + 2q_{2,i} + q_{3,i} = m,$$

where $q_{1,i} \in \{0,\ldots,m\}$, $q_{2,i} \in \{0,1\}$, $q_{3,i} = 0$ if $q_{2,i} = 0$ and $q_{3,i} \in \{0,\ldots,m\}$ if $q_{2,i} = 1$.

The rules for paths starting in node 1 and ending in node 2 are:

1) The number of times edge A_c from node 1 to node 2 is in the path must be 1 larger than the number of times edge A_b is in the path.

2) If edge A_d from node 2 to node 2 is in the path then edge A_c must also be in the path.

3) Edge A_a from node 1 to node 1 can only be in the path after edge A_c, and after itself.

4) Edge A_d from node 2 to node 2 can only be in the path after edge A_b, and after itself.

5) Edge A_c from node 1 to node 2 can only be at the start of the path or after edge A_a or edge A_b.

6) Edge A_b from node 2 to node 1 can only be in the path after edge A_d or edge A_c.

7) The path must end with either edge A_c or A_d.

8) The path must start with either edge A_a or A_c.

9) The number of edges is equal to the matrix power.

These rules result in:

$$A_{2,1,m} = \bigoplus_{(q_1,q_2,q_3,q_4)\in\mathcal{S}_{2,1,m}} \bigotimes_{i=1}^{l} A_\mathrm{d}^{\otimes q_{1,i}} \otimes A_\mathrm{c}^{\otimes q_{2,i}} \otimes A_\mathrm{a}^{\otimes q_{3,i}} \otimes A_\mathrm{b}^{\otimes q_{4,i}}, \tag{3.54}$$

where $q_{1,i} = (q_1)_i$, $q_{2,i} = (q_2)_i$, $q_{3,i} = (q_3)_i$, and where the set $\mathcal{S}_{2,1,m}$ contains all tuples (q_1, q_2, q_3, q_4) that satisfy the following equation:

$$\sum_{i=1}^{l} q_{1,i} + 2q_{2,i} - 1 + q_{3,i} = m,$$

where $q_{1,i} \in \{0, \ldots, m\}$, $q_{2,i} \in \{0, 1\}$, $q_{2,l} = 1$, $q_{3,i} = 0$ if $q_{2,i} = 0$ and $q_{3,i} \in \{0, \ldots, m\}$ if $q_{2,i} = 1$, $q_{4,i} = q_{2,i}$, for $i = 1, \ldots, l-1$, and $q_{4,l} = 0$.

The rules for paths starting in node 2 and ending in node 1 are:

1) The number of times edge A_b from node 2 to node 1 is in the path must be 1 larger than the number of times edge A_c is in the path.
2) If edge A_a from node 2 to node 2 is in the path then edge A_b must also be in the path.
3) Edge A_a from node 1 to node 1 can only be in the path after edge A_c, and after itself.
4) Edge A_d from node 2 to node 2 can only be in the path after edge A_b, and after itself.
5) Edge A_c from node 1 to node 2 can only in the path after edge A_a or edge A_b.
6) Edge A_b from node 2 to node 1 can only be at the start of the path or after edge A_d or edge A_c.
7) The path must end with either edge A_b or A_a.
8) The path must start with either edge A_b or A_d.
9) The number of edges is equal to the matrix power.

These rules result in:

$$[A_0^{\otimes m}]_{1,2} = \bigoplus_{(q_1,q_2,q_3,q_4) \in \mathcal{S}_{1,2,m}} \bigotimes_{i=1}^{l} A_\mathrm{a}^{\otimes q_{1,i}} \otimes A_\mathrm{b}^{\otimes q_{2,i}} \otimes A_\mathrm{d}^{\otimes q_{3,i}} \otimes A_\mathrm{c}^{\otimes q_{4,i}}, \tag{3.55}$$

where $q_{1,i} = (q_1)_i$, $q_{2,i} = (q_2)_i$, $q_{3,i} = (q_3)_i$, and where the set $\mathcal{S}_{1,2,m}$ contains all tuples (q_1, q_2, q_3, q_4) that satisfy the following equation:

$$\sum_{i=1}^{l} q_{1,i} + 2q_{2,i} - 1 + q_{3,i} = m,$$

where $q_{1,i} \in \{0, \ldots, m\}$, $q_{2,i} \in \{0, 1\}$, $q_{2,l} = 1$, $q_{3,i} = 0$ if $q_{2,i} = 0$ and $q_{3,i} \in \{0, \ldots, m\}$ if $q_{2,i} = 1$, $q_{4,i} = q_{2,i}$, for $i = 1, \ldots, l-1$, and $q_{4,l} = 0$.

The same approach can be applied to $\breve{A}_0$ by reordering the state vector as follows: the event times should be split up in departure and arrival times, the first half of the state vector should consist of the departure times, the second half should consist of the arrival times, and both arrival and departure times should be sorted per track. This results in the same structure for matrix $\breve{A}_0$ as A_0; the dimensions of the sub matrices are just larger.

3.5 Reduction of the explicit switching max-plus-linear model

At stations where trains can be reordered, the model can model the change in order of any two trains running over the next track, even if there is a very large time difference between the scheduled departure times. If the maximum of the delays of all trains is known, it is possible to determine which (combinations of) max-plus binary variables will not be used when determining the optimal dispatching actions for reducing the delays. In this section it is explained how these (combinations of) max-plus binary variables can be determined and removed from the model. First, the delay model will be explained. After that the reduction method is explained using the delay model.

3.5.1 Delay model

The model as defined in Section 3.3 has a state vector $x(k)$ that corresponds to the arrival and departure times of the trains. When dealing with delays and trying to minimize them it can be more useful to transform the model such that the transformed state vector $x^{\mathrm{d}}(k)$ shows the delays, with respect to the timetable, instead of the arrival and departure times of the trains. Another advantage of this model transformation is that the elements of the transformed matrix A_μ^{d} are the negative slack times between the events. The concepts of the delay model and negative slack time are based on slack time, realizability, and structural delays as described by Goverde [38, 39]. Define an activity (j,i) as the activity taking place between event j and i connecting event i to event j. In the case of the railway model are the trains running over the tracks, the trains dwelling at the stations, the trains giving connections to other trains, and the signaling system modeled by the headways to keep the trains separated by a safe distance. The slack time is defined in Goverde [38] as:

Definition 3.1 Slack time
For any activity (j,i) the *slack time* is the difference of the end $x_j(k-\mu_{i,j})+a_{i,j}(k,u(k-\mu_{i,j}))$ of the activity and the start $x_i(k)$ of the new activity. □

The slack time can be used to analyze the model with respect to robustness against delays, and in the perturbed mode, it can be used to analyze the effects of the different dispatching actions.

The matrix A_μ is transformed into A_μ^{d}, containing the negative slack times:

$$[A_\mu^{\mathrm{d}}]_{i,j} = [A_\mu]_{i,j} - (r_i(0) - (r_j(0) - \mu T)), \tag{3.56}$$

where we used

$$r(k) = r(0) + kT. \tag{3.57}$$

With these matrices the model from (3.33) can be transformed into

$$x^{\mathrm{d}}(k) = \mathbf{0} \oplus \bigoplus_{\mu=0}^{\mu_{\max}} A_\mu^{\mathrm{d}}(k,u(k-\mu)) \otimes x^{\mathrm{d}}(k-\mu) \oplus \bigoplus_{\mu=-\mu_{\max}}^{-1} A_\mu^{\mathrm{d}}(k,u(k)) \otimes x^{\mathrm{d}}(k-\mu), \tag{3.58}$$

where the state vector $x^{\mathrm{d}}(k)$ contains the delays of the events of cycle k and $\mathbf{0}$ is a vector of the same length as $r(k)$ filled with zeros.

Because the elements of the matrix A_{μ}^{d} represent the negative slack times between events, the value of the elements shows how much one event is delayed directly by another event. For example, if $[A_0^{\mathrm{d}}]_{i,j} = 2$, then $x_i(k)$ is delayed by at least 2 minutes by $x_j(k)$. Therefore, during nominal operation all the elements of the matrices should be negative, since otherwise there would be delays during nominal operation.

3.5.2 Removing redundant control variables

With the use of the delay model and the negative slack times the combinations of max-plus binary variables that cause large delays can be determined. With the use of the following theorem these combinations of max-plus binary variables can be determined.

Theorem 3.1 *The elements of the matrix powers of A_0^d give lower bounds for the delays caused by the max-plus binary variables if and only if the process times used are the minimal process times. The minimal process times are the smallest possible process times that are achievable by the trains and the railway operations.*

Proof: By using the minimum process times, an element of A_0^{d} and can be written as:

$$[A_0^{\mathrm{d}}]_{i,j} = \tau_{i,j}^{\min} + \Delta_{i,j} + u_{i,j} - (r_i(0) - r_j(0)).$$

where $\tau_{i,j}^{\min}$ is a minimum process time(s), $\Delta_{i,j}$ is a positive value equal to the difference between the (sum of) actual process time(s) and the (sum of) minimum process time(s), and $u_{i,j}$ is a max-plus binary variable.

Since $(r_i(0) - r_j(0))$ is fixed by the timetable and $t_{i,j}^{\min}$ is as small as possible, a lower bound for the negative slack time is given for $\Delta_{i,j} = 0$:

$$[A_0^{\mathrm{d}}]_{i,j,\mathrm{lb}} = t_{i,j}^{\min} + u_{i,j} - (r_i(0) - r_j(0)).$$

Since a positive value for $[A_0^{\mathrm{d}}]_{i,j,\mathrm{lb}}$ indicates a delay for x_i, for the $u_{i,j} = 0$, and the minimum process times are used, these values are minimum as well; they are a lower bound to the delay of x_i.

The lower bound on the negative slack time for any element of any power of A_0^{d} can be written as

$$[A_0^{\mathrm{d}}]_{i,j,\mathrm{lb}}^{\otimes p}(k, u(k)) = \bigoplus_{l} (\bigotimes_{q} \tau_{i,j,q,l}^{min}(k) \otimes \bigotimes_{s} u_{i,j,s,l}(k) - (r_i(0) - r_j(0))),$$

where $t_{i,j,q,l}^{\min}(k)$ is a minimum process time and $u_{i,j,s,l}(k)$ is a max-plus binary variable. If the element is independent of max-plus binary variables the max-plus product $\bigotimes_s u_{i,j,s}(k)$ has zero terms. Each element of the max-plus sum is a lower bound for the negative slack term and each of these lower bounds is also a lower bound to the delay for x_i for the combination of $u_{i,j,s,l}(k)$ for which $\bigotimes_s u_{i,j,s,l}(k) = 0$.

□

By using the delay model together with the minimal process times the minimum delays caused by the values of the (combinations of) max-plus binary variables can be determined. Then by assuming there is a known maximum value or upper bound for the delays, the values of the (combinations of) max-plus binary variables that would result in delays larger than the maximum delay, can be identified and the elements of the matrix powers that are non-ε only for the values of those (combinations of) max-plus binary variables can be removed. Removing these elements will have no effect on the solution of the dispatching problem, but it will reduce the computational complexity of the problem.

The values of these (combinations of) max-plus binary variables can be found by determining which values of the max-plus binary variables result in at least one element of the max-plus powers of $\breve{A}_0^{\mathrm{d}}$ being larger than the maximum value for the delays. These values can be determined off-line by making use of Theorem 3.1 and the minimal process times. By using the minimal process times the values of the max-plus powers of $\breve{A}_0^{\mathrm{d}}$ are a lower bound of the delays caused by the different value (combinations of) max-plus binary variables. The reduction is done by replacing the elements that are non-ε only for those values of the (combinations of) max-plus binary variables by ε in all elements of the max-plus powers of $\breve{A}_0^{\mathrm{d}}$, effectively removing them from the model. The reduction can be applied while calculating the feasible explicit model resulting in a reduced explicit model:

$$\breve{x}^{\mathrm{d}}(k) = \breve{A}_0^{\mathrm{d},*,\mathrm{red}} \otimes \left(\breve{\mathbf{0}}(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \breve{A}_{\mu}^{\mathrm{d},\mathrm{red}} \otimes x^{\mathrm{d}}(k-\mu) \right). \tag{3.59}$$

This reduction method can also be applied to the implicit model, but it will not be as effective, because most combinations of max-plus binary variables are only modeled explicitly in the max-plus matrix powers of $\breve{A}_0^{\mathrm{d},*,\mathrm{red}}$ and therefore the value combinations can not be removed from the implicit model. In fact, in the implicit model most combinations of max-plus binary variables are modeled implicitly through the dependency of $x(k)$ on itself.

3.6 Modeling and control of freight trains

In many countries parts of the passenger railways are shared with freight trains. Since freight trains are usually longer and have a heavier load than passenger trains, their dynamics are also quite different: they accelerate and decelerate much slower; especially starting up again after a stop takes a long time. Therefore, changes in the running time and unscheduled stops have much larger effects on the headway times of freight trains compared to passenger trains and this relation between headway times and running times and unscheduled stops must be modeled and if possible avoided.

To be able to model and control the running times and stops of the freight trains we need to make changes to the switching max-plus-linear model and add extra control constraints to the model predictive controller. First we will describe the changes to the model to include unscheduled stops at stations and reduced speed or full stops on open track for freight trains. After that we give the constraints for the model predictive

controller.

3.6.1 Unscheduled stop at a station

At some points in the network, such as stations and special overtaking tracks, the freight train can make unscheduled stops to allow other trains to overtake it. This is a dispatching action that needs to be modeled. To be able to model this the running time, continuity, and the headway constraints need to be adjusted. The running time constraints need to be changed since an unscheduled stop will increase the running time of the train run before and after the unscheduled stop, because of the deceleration and acceleration. This increase can be modeled by adjusting the running time constraint. Consider a freight train with train runs p_i and i that has no scheduled stop between train run p_i and i, but it can stop there. This is then modeled by:

$$a_{p_i}(k) \geq d_{p_i}(k) \otimes \tau_{\mathrm{r},p_i}(k) \otimes (0 \oplus \tau_{\mathrm{r},p_i,2}(k) \otimes \overline{u_{\mathrm{d},p_i,2}(k)}) \tag{3.60}$$

$$a_i(k) \geq d_i(k) \otimes \tau_{\mathrm{r},i}(k) \otimes (0 \oplus \tau_{\mathrm{r},i,2}(k) \otimes \overline{u_{\mathrm{d},p_i,2}(k)}), \tag{3.61}$$

where $u_{\mathrm{d},p_i,2} \in \{0, \varepsilon\}$ is a max-plus binary variable that determines whether the freight train makes an unscheduled stop or not. If the max-plus binary variable is ε the extra running times $\tau_{\mathrm{r},p_i,2}(k)$ and $\tau_{\mathrm{r},i,2}(k)$ are added; if it is 0 the normal running time is used.

The continuity constraint between p_i and i has to be modified as well to model the increased dwell time:

$$d_i(k) \geq a_{p_i}(k) \otimes \tau_{\mathrm{d},i,p_i}(k) \otimes (0 \oplus \tau_{\mathrm{d},i,p_i,2}(k) \otimes \overline{u_{\mathrm{d},p_i,2}(k)}), \tag{3.62}$$

where $\tau_{\mathrm{d},i,p_i,2}(k)$ is the minimum increase in dwell time due to the stop.

The increase in running time and the deceleration at the end of train run $p_i(k)$ also have an effect on the arrival headway times between the trains traversing the track after the train of train run $p_i(k)$. For simplicity let the train of train run $p_i(k)$ be the first train to traverse the track and let $\mathcal{H}_{p_i}$ be the set of train runs on the same track as $p_i(k)$ that start after train run $p_i(k)$. The headway constraints can then be written as:

$$d_j(k) \geq d_{p_i}(k) \otimes \tau_{\mathrm{h,d},j,p_i}(k) \tag{3.63}$$

$$a_j(k) \geq a_{p_i}(k) \otimes \tau_{\mathrm{h,a},j,p_i}(k) \otimes (0 \oplus \tau_{\mathrm{h,a},j,p_i,2}(k) \otimes \overline{u_{\mathrm{d},p_i,2}(k)}), \tag{3.64}$$

for $j \in \mathcal{H}_{p_i}$, and where $\tau_{\mathrm{h,a},j,p_i}(k)$ is the default headway time and $\tau_{\mathrm{h,a},j,p_i}(k) + \tau_{\mathrm{h,a},j,p_i,2}(k)$ is the headway time when the train of train run $p_i(k)$ makes an unscheduled stop after train run $p_i(k)$.

The departure headway times of the trains traversing the track after train run $i(k)$ are also increased if the freight train has an unscheduled stop before train run $i(k)$. Let $\mathcal{H}_i$ be the set of train runs on the same track as $i(k)$ that start after train run $i(k)$. The headway constraints can then be written as:

$$d_j(k) \geq d_i(k) \otimes \tau_{\mathrm{h,d},j,i}(k) \otimes (0 \oplus \tau_{\mathrm{h,d},j,i,2}(k) \otimes \overline{u_{\mathrm{d},p_i,2}(k)}) \tag{3.65}$$

$$a_j(k) \geq a_i(k) \otimes \tau_{\mathrm{h,a},j,i}(k), \tag{3.66}$$

for $j \in \mathcal{H}_{p_i}$, and where $\tau_{\mathrm{h,d},j,i}(k)$ is the default headway time and $\tau_{\mathrm{h,d},j,i}(k) + \tau_{\mathrm{h,d},j,i,2}(k)$ is the headway time when the train of train run $i(k)$ has made an unscheduled stop at the station before $i(k)$.

The headway constraints can be extended to allow the reordering of the trains on the tracks as described in Section 3.3.

3.6.2 Reduced speed/full stop on open track

Besides making stops at stations or at special overtaking tracks in the network it can happen that a freight train has to slow down to avoid a yellow or red signal light. Slowing down the freight train has a large impact on the headway times. Furthermore it takes much longer for freight trains to slow down and speed up to the maximum allowed velocity. We should therefore try to having freight trains run into yellow or red signal lights. To model this we include additional process times that depend on binary variables into the running and headway constraint. For the running time constraints this results in:

$$a_{p_i}(k) \geq d_{p_i}(k) \otimes \tau_{\mathrm{r},p_i}(k) \otimes \bigotimes_{l} (0 \oplus \tau_{\mathrm{r},p_i,l}(k) \otimes \overline{u_{\mathrm{r},p_i,l}(k)}), \tag{3.67}$$

for $l = 1, \ldots$. Different max-plus binary variables $u_{\mathrm{r},p_i,l}$ increase the running time with different amounts denoted by $\tau_{\mathrm{r},p_i,p_i,l}(k)$, allowing control of the increase of running time. The number of terms l in the sum is the choice of the designer. Having more terms in the sum allows for more choices in the length of the running times, but also increases the number of max-plus binary variables of the model, which will make determining the optimal dispatching actions more computationally complex.

For the headway constraints the adjustment results in:

$$a_j(k) \geq a_{p_i}(k) \otimes \tau_{\mathrm{h,a},j,p_i}(k) \otimes \bigotimes_{l} (0 \oplus \tau_{\mathrm{h,a},j,p_i,l}(k) \otimes \overline{u_{\mathrm{r},p_i,l}(k)}) \tag{3.68}$$

$$d_j(k) \geq d_{p_i}(k) \otimes \tau_{\mathrm{h,d},j,p_i}(k) \otimes \bigotimes_{l} (0 \oplus \tau_{\mathrm{h,d},j,p_i,l}(k) \otimes \overline{u_{\mathrm{r},p_i,l}(k)}), \tag{3.69}$$

where $\tau_{\mathrm{h,a},j,p_i,l}$ and $\tau_{\mathrm{h,d},j,p_i,l}$ are the increased headway times for the arrivals and departures due to the decreased velocity of the trains on the track.

3.6.3 Optimization constraints for freight trains

With the SMPL model the events take place as soon as all constraints are satisfied, this also means that the defined process times are just minimal times and during the calculation of the dispatching actions and new timetable there is no restriction on the length of the process times, even without changing the max-plus binary variables. Constraints must be added to give upper bounds on the process times based on the values of the max-plus binary variables. These constraints ensure that during the calculation of the dispatching actions and new timetable the max-plus binary variables are used when the running and dwell times need to be increased. Furthermore these constraints do not model any dynamics of the railway traffic or network, and therefore they are not part of the SMPL model, but are optimization constraints. Therefore these constraints are described as mixed-integer-linear equations instead of max-plus-linear equations.

For the unscheduled stop at a station or special overtaking track the constraints to limit the running times are given by:

$$a_{p_i}(k) \leq d_{p_i}(k) + \tau_{\mathrm{r},p_i}(k) + \tau_{\mathrm{r},p_i,2}(k)\upsilon_{\mathrm{d},p_i,2}(k) + \tau_{\mathrm{r},p_i,3}(k) \tag{3.70}$$

$$a_i(k) \leq d_i(k) + \tau_{\mathrm{r},i}(k) + \tau_{\mathrm{r},i,2}(k)\upsilon_{\mathrm{d},p_i,2}(k) + \tau_{\mathrm{r},i,3}(k), \tag{3.71}$$

where $\upsilon_{\mathrm{d},p_i,2}(k) = 0$ if $u_{\mathrm{d},p_i,2}(k) = 0$ and $\upsilon_{\mathrm{d},p_i,2}(k) = 1$ if $u_{\mathrm{d},p_i,2}(k) = \varepsilon$, and $\tau_{\mathrm{r},i,3}(k)$ is the maximum increase in running time before the headway times need to be adjusted. These constraints ensure that the running time for train run p_i is between $\tau_{\mathrm{r},p_i}(k)$ and $\tau_{\mathrm{r},p_i}(k) + \tau_{\mathrm{r},p_i,3}(k)$ if $u_{\mathrm{d},p_i,2}(k) = \varepsilon$ and between $\tau_{\mathrm{r},p_i}(k) + \tau_{\mathrm{r},p_i,2}(k)$ and $\tau_{\mathrm{r},p_i}(k) + \tau_{\mathrm{r},p_i,2}(k) + \tau_{\mathrm{r},p_i,3}(k)$ if $u_{\mathrm{d},p_i,2}(k) = 0$.

The constraint to limit the dwell time in case of an unscheduled stop at a station or special overtaking track is given by:

$$d_i(k) \leq a_{p_i}(k) + \tau_{\mathrm{d},i,p_i}(k) + \tau_{\mathrm{d},i,p_i,3}(k) - \beta\upsilon_{\mathrm{d},p_i,2}, \tag{3.72}$$

where $\upsilon_{\mathrm{d},p_i,2}$ is defined as before, $\beta \ll 0$ is a large negative value, and $\tau_{\mathrm{d},i,p_i}(k) + \tau_{\mathrm{d},i,p_i,3}(k)$ is the upper limit to the dwell time if the freight train does not stop. If it does stop, $\upsilon_{\mathrm{d},p_i,2}$ is set to 1 and a large positive value $-\beta$ is added effectively removing the upper bound on the dwell time.

In the case of breaking and/or stopping because of a signal light the constraint for the upper limit on the running time constraint is given by:

$$a_{p_i}(k) \leq d_{p_i}(k) + \tau_{\mathrm{r},p_i,1}(k) + \tau_{\mathrm{r},p_i,3}(k) + \sum_l \tau_{\mathrm{r},p_i,l}(k)\upsilon_{\mathrm{r},p_i,l}, \tag{3.73}$$

where l is the same as in (3.67) and is chosen by the designer, $\upsilon_{\mathrm{r},p_i,l}(k) = 0$ if $u_{\mathrm{r},p_i,l}(k) = 0$ and $\upsilon_{\mathrm{r},p_i,l}(k) = 1$ if $u_{\mathrm{r},p_i,l}(k) = \varepsilon$, $\tau_{\mathrm{r},p_i,3}(k)$ is the maximum increase in running time without needing to increase the headway times, and the upper limit on the running time is given by $\tau_{\mathrm{r},p_i,1}(k) + \tau_{\mathrm{r},p_i,3}(k) + \sum_l \tau_{\mathrm{r},p_i,p_i,l}(k)\upsilon_{\mathrm{r},p_i,l}$.

The headway constraints do not need upper limits; during the determination of the dispatching actions and the new timetable, the headway times may be longer than the minimum headway times, that just means the trains are traversing the tracks further apart than strictly required and there is still some buffer time between the trains.

3.6.4 Example

As an example consider a freight train and a passenger train on a small network three stations and two tracks connecting the three stations. The trains only drive in one direction and do not continue after the third station. The freight train can make an unscheduled stop at the station connecting the two tracks. On both tracks the order of the trains can be changed. On both tracks the freight train can run into signal lights and may have to slow down or stop.

Let the freight train on the first track be denoted by train run 1 and on the second track by train run 3. Let the passenger train on the first track be denoted by train run 2 and on the second track by train run 4.

First we will describe the running time constraints on the first track:

$$a_1(k) \geq d_1(k) \otimes \tau_{\mathrm{r},1}(k) \otimes (0 \oplus \tau_{\mathrm{r},1,2}(k) \otimes \overline{u_{\mathrm{d},1,2}(k)}) \otimes \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{r},1,l}(k) \otimes \overline{u_{\mathrm{r},1,l}(k)})$$

$$a_2(k) \geq d_2(k) \otimes \tau_{\mathrm{r},2}(k).$$

The running time constraints have max-plus binary variable $u_{\mathrm{d},1,2}(k)$ for planning an unscheduled stop at the station connecting the two tracks, and for the reduced speed and full stop at track 1 the max-plus binary variables $u_{\mathrm{r},1,l}(k)$, for $l = 1\ldots,b$, where b can be chosen by the freely and determines the number of possible steps in the speed reduction that are modeled.

The linear optimization constraint that ensures the running time of the freight train cannot increase without changing the max-plus binary variables is given by:

$$a_1(k) \leq d_1(k) + \tau_{\mathrm{r},1}(k) + \tau_{\mathrm{r},1,2}(k)\upsilon_{\mathrm{d},1,2}(k) + \tau_{\mathrm{r},1,3}(k) + \sum_{l=3\ldots,b} \tau_{\mathrm{r},1,l}(k)\upsilon_{\mathrm{r},p_i,l},$$

where $\upsilon_{\mathrm{d},1,2}(k) = 0$ if $u_{\mathrm{d},1,2}(k) = 0$ and $\upsilon_{\mathrm{d},1,2}(k) = 1$ if $u_{\mathrm{d},1,2}(k) = \varepsilon$, and $\upsilon_{\mathrm{d},1,l}(k) = 0$ if $u_{\mathrm{d},1,l}(k) = 0$ and $\upsilon_{\mathrm{d},1,l}(k) = 1$ if $u_{\mathrm{d},1,l}(k) = \varepsilon$.

The headway constraints on the first track are:

$$\begin{aligned}
d_2 \geq & d_1 \otimes \tau_{\mathrm{h,d},2,1}(k) \otimes (0 \oplus \tau_{\mathrm{h,d},2,1,2}(k) \otimes \overline{u_{\mathrm{d},1,2}(k)}) \otimes u_{2,1}(k) \otimes \\
& \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{h,d},2,1,l}(k) \otimes \overline{u_{\mathrm{r},1,l}(k)}) \\
d_1 \geq & d_2 \otimes \tau_{\mathrm{h,d},1,2}(k) \otimes \overline{u_{2,1}(k)} \\
a_2 \geq & a_1 \otimes \tau_{\mathrm{h,a},2,1}(k) \otimes u_{2,1}(k) \otimes \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{h,a},2,1,l}(k) \otimes \overline{u_{\mathrm{r},1,l}(k)}) \\
a_1 \geq & a_2 \otimes \tau_{\mathrm{h,a},1,2}(k) \otimes \overline{u_{2,1}(k)}.
\end{aligned}$$

The headway constraints include max-plus binary variable $u_{2,1}(k)$ for changing the order of the trains, $u_{\mathrm{d},1,2}(k)$ for planning an unscheduled stop at the station connecting the two tracks, and variables $u_{\mathrm{r},1,l}(k)$, for $l = 1\ldots,b$, for the reduced speed and full stop at track 1.

The continuity constraints on the station connecting the two tracks are given by:

$$\begin{aligned}
d_3(k) \geq & a_1(k) \otimes \tau_{\mathrm{d},3,1}(k) \otimes (0 \oplus \tau_{\mathrm{d},3,1,2}(k) \otimes \overline{u_{\mathrm{d},1,2}(k)}) \\
d_4(k) \geq & a_2(k) \otimes \tau_{\mathrm{d},4,2}(k).
\end{aligned}$$

The constraint to limit the dwell time of the freight train at the station is given by:

$$d_3(k) \leq a_1(k) + \tau_{\mathrm{d},3,1}(k) + \tau_{\mathrm{d},3,1,3}(k) - \beta\upsilon_{\mathrm{d},1,2},$$

The running time constraints on the second track are similar to those of the first track, except that the freight train is already scheduled to stop at the third station, so no unscheduled stop can be planned:

$$\begin{aligned}
a_3(k) \geq & d_3(k) \otimes \tau_{\mathrm{r},3}(k) \otimes \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{r},3,l}(k) \otimes \overline{u_{\mathrm{r},3,l}(k)}) \\
a_4(k) \geq & d_4(k) \otimes \tau_{\mathrm{r},4}(k).
\end{aligned}$$

The headway constraints on the second track are:

$$
\begin{aligned}
d_4 \geq & d_3 \otimes \tau_{\mathrm{h,d},4,3}(k) \otimes u_{4,3}(k) \otimes \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{h,d},4,3,l}(k) \otimes \overline{u_{\mathrm{r},3,l}(k)}) \\
d_3 \geq & d_4 \otimes \tau_{\mathrm{h,d},3,4}(k) \otimes \overline{u_{4,3}(k)} \\
a_4 \geq & a_3 \otimes \tau_{\mathrm{h,a},4,3}(k) \otimes u_{4,3}(k) \otimes \bigotimes_{l=3\ldots,b} (0 \oplus \tau_{\mathrm{h,a},4,3,l}(k) \otimes \overline{u_{\mathrm{r},3,l}(k)}) \\
a_3 \geq & a_4 \otimes \tau_{\mathrm{h,a},3,4}(k) \otimes \overline{u_{4,3}(k)}.
\end{aligned}
$$

There are no continuity constraints for the trains at station 3 since we assume they do not continue after arriving there.

3.7 Summary

In this chapter it has been shown how railway traffic for passenger railways that is driving according to a given schedule, can be modeled as max-plus-linear models. By using max-plus binary variables the max-plus-linear model has been extended to a switching max-plus-linear model. The switching max-plus-linear model can model the effects of different dispatching actions such as reordering of trains on a track, switching between parallel tracks between stations, and breaking connections. Furthermore, it has been shown how an implicit SMPL model of the railway traffic can be converted into its explicit form for single and multiple cycles. Finally an extension to the model for passenger railways was introduced such that freight trains can also be modeled and controlled. This includes new dispatching actions for unscheduled stops of freight trains and for increasing running times.

Chapter 4

Model predictive control for railway traffic management

This chapter introduces the concepts of model predictive control (MPC) and distributed model predictive control (DMPC) and gives an overview of the history of (D)MPC. We show how MPC can be used to solve the problem of railway traffic management in Section 4.2. In Section 4.3 we propose four DMPC methods. In Sections 4.4 and 4.5 we perform various case studies to determine the effectiveness of the proposed MPC and DMPC methods. In Section 4.6 we draw conclusions and summarize the results. The main contributions of this chapter are the proposed MPC and DMPC methods for railway traffic management and the case studies that were performed.

Parts of this chapter have been published in the papers by Kersbergen et al. [57, 58, 59].

4.1 Introduction

In this chapter model predictive control (MPC), in particular for the application of railway traffic management, will be explained. But before the details of the application are explained, the basics of MPC will be discussed. MPC is a control methodology that, at discrete time instants $t(\kappa)$ for $\kappa = 0, 1, \ldots$ determines the control inputs for the system that minimize a cost function or maximize a performance criterion, based on a prediction of the evolution of the state of the system under control. The prediction is done using a model of the system under control. MPC can be characterized by five components:

- A model of the system (and of disturbances).
  The model is used to predict the effects of the control inputs on the evolution of the system over a given prediction horizon and to determine the control actions that minimize or maximize a given performance index. The disturbances are also modeled to predict the effects of the disturbances on the system and to minimize them.

- A cost function (performance critter)
  The cost function (performance critter) is a function of the predicted state evolution, a reference signal and the control inputs. It is used as a measure to evaluate the quality of the control actions and is chosen by the designer of the controller.

- Constraints.
  One of the differentiating characteristics of MPC is that it can directly handle constraints. These constraints can be on the input, the output, or the state of the system, e.g. to limit the rate of change in the output, or to limit the range or shape of the input.

- Optimization.
  At time instants $t(\kappa)$ the controller receives information about the system, usually the output, and it uses the information and a model of the system under control to determine the control input(s) over the control horizon, that minimize the cost function. To determine the control inputs that minimize the performance index each time instant an optimization problem is built using the model,cost function, and the constraints. Depending on the type of model (e.g. linear, non-linear, discrete event) and the cost function (e.g. linear, quadratic, 1-norm, ∞-norm) the resulting optimization can be a (mixed integer) linear, non-linear, or quadratic programming problem.

  Depending on the system the optimization problem can be very hard to solve. To limit the size of the optimization problem a finite prediction horizon is chosen. The computational complexity of the optimization can be further reduced by reducing the length prediction horizon. A second possibility to reduce the computational complexity is the control horizon. The controller predicts the evolution of the system over the whole prediction horizon but is only allowed to change the control input during the control horizon. The control horizon is from $t(\kappa)$ till $t(\kappa)+c\tau_{\mathrm{H}}$, where τ_{H} is the step size and c is the number of steps on the control horizon. The prediction horizon is from $t(\kappa)$ till $t(\kappa)+p\tau_{\mathrm{H}}$, where p is the number of steps in the prediction horizon.

- A receding horizon.
  A receding horizon means that once the controller has determined the control input(s) at time instant $t(\kappa)$ it only implements the control input(s) in the interval $[t(\kappa), t(\kappa)+\tau_{\mathrm{H}})$ and the next time instant $t(\kappa+1)$ it shifts the prediction and control horizon of the optimization problem to $[t(\kappa+1), t(\kappa+1)+p\tau_{\mathrm{H}})$ and $[t(\kappa+1), t(\kappa+1)+c\tau_{\mathrm{H}})$ respectively, and repeats the optimization described above. It repeats this process of implementing only the first part of the control inputs, shifting the horizon and recomputing the control actions using the optimization at every time instant.

The time between two time instants may vary depending on the system. In the case of continuous-time dynamical models the time between two time instants may be a fixed time τ_{H}. We call this time-based MPC. For discrete-event systems another possible definition

for the time between two instants is the time it takes for a number of events to occur after the current time instant $t(\kappa)$. When this is the case we call it event-based MPC. Another possibility is that the time between two time instants is defined by the time it takes for the system performance to drop below a given threshold, or for the model error to become larger than a given maximum, usually this time is bounded between a minimum time $\tau_{\min}$ and a maximum time $\alpha\tau_{\min}$ ($\alpha > 1$). The time between two instant may also be determined by the time it takes for new information from past events or measurements of the system to become available.

MPC for linear models was first described and successfully applied in the late 1970s and early 1980s by Richalet et al. [75] and Cutler and Ramaker [20] in the chemical process industry. In the next decades MPC was applied successfully many more times and extended to include non-linear models (see [32, 71, 73] and the references therein). In the early 2000s a further extension of MPC to max-plus-linear discrete-event systems was made by De Schutter and van den Boom [25, 26].

As was discussed in the introduction many papers have been published on the development of railway traffic management systems. In almost all of these papers some form of MPC is used. But most of these methods only consider a part of the network and almost none of them consider how the whole network can be controlled. Expanding the current methods to consider the whole network is one possibility, but then most of these methods will run in computation problems. Another possibility is the approach with a supervisory controller such as the one by Corman et al. [13, 14, 16], but then solving the supervisory problem may become very hard for a large number of subnetworks. A third possibility is distributed model predictive control (DMPC) [9, 29, 31, 76]. With DMPC there is no supervisory controller: instead the model predictive controllers solving the dispatching problems for the subnetworks communicate and coordinate with each other to reach a global feasible solution. For an overview of DMPC methods the interested reader is referred to [69].

In Section 4.2 we describe the basics of MPC for on-line railway traffic management and how the optimization problem can be built up with the use of the model described in Chapter 3. The DMPC methods we propose are given in Section 4.3. The next two sections consist of two case studies. In Section 4.4 we test the performance of the model predictive controllers using the implicit and explicit models of the previous chapter. In Section 4.5 we compare the performance of the DMPC methods to the centralized MPC method. Finally in Section 4.6 we summarize the chapter.

4.2 MPC for on-line railway traffic management

We will use time-based model predictive control, with a fixed step size for the time instants, to do the rescheduling of the railway traffic. The reason for using time-based MPC is that the time between two time instants is fixed, where as for event-based MPC it varies between time instants, depending on how much time there is between events. The biggest challenge of on-line railway traffic management is determining the new schedule in a short

amount of time. If the maximum time to compute the solution is not known since the time between time instants constantly varies, it is hard to ensure that the solution is found in time. It is easier to do when the time between time instants is fixed and known, then a clear bound on the computation time for the optimization can be given. The model used in this case is the SMPL model described in the previous chapter and given in (3.49) or (3.50) for the implicit and explicit model respectively.

4.2.1 Prediction and control horizon at time instant $t(\kappa)$

Recall that the time instants are denoted by $t(\kappa)$, for $\kappa = 0, 1, \ldots$, where κ is the time instant counter. Since we use time-based model predictive control with a fixed time step $t(\kappa+1) = t(\kappa) + \tau_{\mathrm{H}}$, where τ_{H} is the step size. To be able to describe the control problem at time instant $t(\kappa)$ we first need to define the prediction and control horizon. After that we explain how the events and control inputs that are in the prediction and control horizon can be determined for each time instant $t(\kappa)$ and how that information is used to determine the part of the model that is needed for the optimization at $t(\kappa)$.

The prediction horizon determines the events that need to be considered and for which the event time must be determined. Let the length of the prediction horizon be defined by the number of time steps, denoted by p, then the prediction horizon is defined to be from $t(\kappa)$ till $t(\kappa) + p\tau_{\mathrm{H}}$.

The control horizon determines for which events the binary variables can be adjusted. The length of the control horizon is defined by the number of time steps and is denoted by c, with $c \leq p$. The control horizon is then defined to be from $t(\kappa)$ till $t(\kappa) + p\tau_{\mathrm{H}}$.

4.2.2 Events and control variables at time instant $t(\kappa)$

Let $\mathcal{X}(\kappa)$ contain all event times in the extended state vector $\breve{x}(k)$ from (3.49) or (3.50) that have not yet occurred and are predicted at $t(\kappa)$ or scheduled according to the original timetable to occur in $[t(\kappa), t(\kappa) + p\tau_{\mathrm{H}})$:

$$\breve{x}_i(k) \in \mathcal{X}(\kappa) \text{ if } \begin{cases} t(\kappa) \leq \breve{x}_i(k) < t(\kappa) + p\tau_{\mathrm{H}} \\ \text{or} \\ t(\kappa) \leq \breve{x}_i(k) \text{ and } \breve{r}_i(k) < t(\kappa) + p\tau_{\mathrm{H}} \end{cases}, \tag{4.1}$$

and let $\chi(\kappa)$ be a vector containing all elements of $X(\kappa)$.

Not all max-plus binary variables can be changed at each time instant. To determine the max-plus binary variables that can be changed at $t(\kappa)$ consider the SMPL model in (3.49). Each element of the matrices of the SMPL model relates two events to each other. If the value of that element depends on a max-plus binary variable and if at least one of the events associated to it is in the control horizon and the other is in the control or prediction horizon that max-plus binary variable can be changed at the current time instant and is in $\psi(\kappa)$. Let $\breve{u}(k)$ be a vector containing all max-plus binary variables of the system in (3.49). Each max-plus binary variable $\breve{u}_i(k)$ is related to one or more variables in $\breve{x}(k)$ through the constraints in Section 3.3. Let $\mathcal{X}_{\breve{u}_i(k)}$ be the set containing the continuous variables in $\breve{x}(k)$ related to $\breve{u}_i(k)$, then the set $\mathcal{U}(\kappa)$ contains all max-plus binary variables that can be changed by the model predictive controller at time instant

$t(\kappa)$, resulting in

$$\breve{u}_i(k) \in \mathcal{U}(\kappa) \text{ if } \begin{cases} t(\kappa) \leq \breve{x}_j(k) < t(\kappa) + c\tau_{\mathrm{H}} \text{ for } \exists \breve{x}_j(k) \in \mathcal{X}_{\breve{u}_i(k)} \\ \text{and} \\ \breve{x}_j(k) \in \chi(\kappa) \text{ for } \forall \breve{x}_j(k) \in \mathcal{X}_{\breve{u}_i(k)} \end{cases}, \tag{4.2}$$

and let $\psi(\kappa)$ be the vector containing all elements of $\mathcal{U}(\kappa)$.

All events, and their associated control variables that occurred before $t(\kappa)$ are assumed to be in the past and their event times are assumed to be known. The max-plus binary variables that are only associated to the events that are predicted at $t(\kappa)$ or scheduled according to the original timetable to occur in $[t(\kappa)+c\tau_{\mathrm{H}}, t(\kappa)+p\tau_{\mathrm{H}})$ are set to their nominal value, which is zero[1].

4.2.3 Model at time instant $t(\kappa)$

The part of the SMPL model needed to determine the event times in $\chi(\kappa)$ is extracted from the complete model and will be converted into a set of mixed integer linear constraints. This process is shown in Section 4.2.6. By describing the SMPL model as a set of mixed-integer-linear constraints and choosing a linear cost function the optimization problem can be solved as a mixed integer linear programming (MILP) problem.

In order to convert the SMPL model into a set of mixed-integer-linear constraints the max-plus binary control variables will be modeled by regular binary variables $\upsilon_i(\kappa) \in \{0,1\}$. This can be done by multiplying the normal binary variables by a large negative number $\beta \ll 0$:

$$\psi_i(\kappa) \approx \upsilon_i(\kappa)\beta \qquad \overline{\psi_i(\kappa)} \approx (1-\upsilon_i(\kappa))\beta, \tag{4.3}$$

where β must be a sufficiently large negative number such that constraints containing $\upsilon_i\beta$ are always satisfied when $\upsilon_i = 1$.

4.2.4 Cost function at $t(\kappa)$

Before the possible cost functions are proposed, first the definition for delay should be given:

Definition 4.1 Delay
For any event $x_i(k)$ with a scheduled event time $r_i(k)$ the delay is defined as the deviation from its scheduled event time $r_i(k)$:

$$x_i^{\mathrm{d}}(k) = x_i(k) - r_i(k), \tag{4.4}$$

where $x_i^{\mathrm{d}}(k)$ is the delay of event $x_i(k)$. □

Since all of these events have a scheduled event time, they also have a timetable constraint, therefore $x_i(k) \geq r_i(k)$ and $x_i^{\mathrm{d}}(k) \geq 0$. Next several vectors need to be defined. Define $\upsilon(\kappa)$

[1] When the control variables are zero the trains run over the tracks in the nominal order, all connections are maintained, and all trains run on the tracks they were originally planned on; only the arrival and departure times are adjusted to avoid conflicts.

as the vector containing all binary variables associated to the max-plus binary variables in $\psi(\kappa)$. Define $\iota(\kappa)$ as the vector containing the scheduled arrival or departure times of the events in the prediction horizon at $t(\kappa)$. Define $\chi^{\mathrm{d}}(\kappa)$ as the delays of the events in $\chi(\kappa)$. Finally define $\mathbf{1}^{1\times m}$ as a 1 by m vector containing only ones. With these definition the cost function can be defined.

For railway traffic management the goal is almost always to minimize the delays; so it makes sense that the performance criterion reflects this. Obvious choices for the performance criterion would be the minimization of the sum of all delays, or the sum of arrival delays, or minimizing the passenger delays if detailed passenger information is available. The sum of all delays would translate to the following cost function of the optimization:

$$\begin{aligned} J(\kappa) &= \mathbf{1}^{1\times n_\chi(\kappa)} \cdot \chi^{\mathrm{d}}(\kappa) + \varrho \mathbf{1}^{1\times n_\upsilon(\kappa)} \cdot \upsilon(\kappa) \\ &= \mathbf{1}^{1\times n_\chi(\kappa)} \cdot (\chi(\kappa) - \iota(\kappa)) + \varrho \mathbf{1}^{1\times n_\upsilon(\kappa)} \cdot \upsilon(\kappa) \\ &= \mathbf{1}^{1\times n_\chi(\kappa)} \cdot \chi(\kappa) + \varrho \mathbf{1}^{\times n_\upsilon(\kappa)} \cdot \upsilon(\kappa) - \mathbf{1}^{1\times n_\chi(\kappa)} \cdot \iota(\kappa), \end{aligned}$$

where ϱ is a small positive value used to ensure that the minimum number of changes are made to the schedule when multiple solution result in the same value for the sum of delays, $n_\chi(\kappa)$ is the number of event times in $\chi(\kappa)$, and n_υ is the number of binary variables in $\upsilon(\kappa)$. The preference for solutions with minimal changes compared to the schedule is based on the idea that at some point in the future all trains should run according to the schedule again and that means that any change made to the schedule must also be reversed at some point, therefore we prefer solutions with less changes if the delays are the same for the solutions. Note that $\mathbf{1}^{1\times n_\chi(\kappa)} \cdot \iota(\kappa)$ is a constant value and the goal is to minimize the cost function, there the constant term does not influence the solution and can be removed from the cost function. This means that minimizing the sum of delays is the same as minimizing the sum of event times.

For the sum of arrival delays this results in the cost function:

$$J(\kappa) = [\mathbf{1}^{1\times n_a(\kappa)}\ \mathbf{0}^{1\times n_d(\kappa)}] \cdot \left(\begin{bmatrix} \chi_{\mathrm{a}}(\kappa) \\ \chi_{\mathrm{d}}(\kappa) \end{bmatrix} - \begin{bmatrix} \iota_{\mathrm{a}}(\kappa) \\ \iota_{\mathrm{d}}(\kappa) \end{bmatrix} \right) + \varrho \mathbf{1}^{1\times n_\upsilon(\kappa)} \cdot \upsilon(\kappa),$$

where $\mathbf{0}^{1\times n_d(\kappa)}$ is a vector containing only zeros of length $n_{\mathrm{d}}(\kappa)$, $n_{\mathrm{d}}(\kappa)$ is the number of departure times in $\chi(\kappa)$, $n_{\mathrm{a}}(\kappa)$ is the number of arrival times in $\chi(\kappa)$, $\chi_{\mathrm{a}}(\kappa)$ is the vector containing the arrival times, $\chi_{\mathrm{d}}(\kappa)$ is the vector containing the departure times, and $\iota_{\mathrm{a}}(\kappa)$, $\iota_{\mathrm{d}}(\kappa)$ contain the scheduled arrival and departure times respectively. For the sum of departure delays the zero and one vector in the cost function should be switched.

Another possibility is to minimize the maximum (arrival/departure) delay. For this a new continuous variable $\chi_{\max}$ needs to be added to the optimization problem that is the maximum of all (arrival/departure) delays. This can be achieved quite easily by adding a set of constraints for the maximum of all delays:

$$\chi_{\max} \geq \chi_i(\kappa) - \iota_i(\kappa) \quad \text{for } \forall \chi_i(\kappa) \in \chi(\kappa).$$

For the arrival delays the set of constraints would be

$$\chi_{\max} \geq \chi_{\mathrm{a},i}(\kappa) - \iota_{\mathrm{a},i}(\kappa) \quad \text{for } \forall \chi_{\mathrm{a},i}(\kappa) \in \chi(\kappa).$$

For the departure delays the set of constraints would be:

$$\chi_{\max} \geq \chi_{\mathrm{d},i}(\kappa) - \iota_{\mathrm{a},i}(\kappa) \quad \text{for } \forall \chi_{\mathrm{d},i}(\kappa) \in \chi(\kappa).$$

The cost function would then be:

$$J(\kappa) = \chi_{\max} + \varrho \mathbf{1}^{1 \times n_{\upsilon}(\kappa)} \cdot \upsilon(\kappa).$$

All of these cost functions are based on the delays of trains, but for the passengers it would be better to focus on the delays the passengers get. To be able to do that a lot of information on the passengers and their behavior would be needed. With the recent introduction of smart cards to pay for the railway tickets, railway operators gained a new source of information to estimate the number of passengers in the trains and several researchers have been working on using this data to estimate the passenger flows [2, 61, 87]. With this research the cost functions can be changed from a measure of train delays to a measure of passenger delays. As future research changing the cost function from a measure of train delays to a measure of passenger delays would be very interesting, but is outside the scope of this thesis.

4.2.5 Optimization at time instant $t(\kappa)$

With the partial SMPL model converted into a set of mixed-integer-linear constraints and the cost function defined the optimization problem can be written as:

$$\min_{z(\kappa)} c^{\top}(\kappa)\, z(\kappa) \tag{4.5}$$

$$\text{s.t. } A(\kappa)\, z(\kappa) \leq b(\kappa), \tag{4.6}$$

where

$$z(\kappa) = \begin{bmatrix} \chi(\kappa) \\ \upsilon(\kappa) \end{bmatrix}.$$

Equation (4.5) contains the cost function that needs to be minimized, where $c(\kappa)$ is a weighting vector, and (4.6) contains the mixed-integer-linear constraints.

4.2.6 Example

To illustrate the whole process consider an SMPL model of three trains traversing a track every half hour and assume that all headway times are 3 minutes. To keep the example simple there will be no delays in this scenario. The scheduled departure, arrival, and minimum running times are given in Table 4.1.

This results in the following timetable vector:

$$r(0) = \begin{bmatrix} 0 & 5 & 10 & 20 & 25 & 28 \end{bmatrix}^{\top}.$$

For the simplicity of the example we assume trains can only change order in the next, the same or the previous cycle, so $\mu_{\max} = 1$.

Table 4.1: Timetable for the three trains of the example.

Train	Departure (min)	Arrival (min)	Running time (min)
1	0	20	18
2	5	25	18
3	10	28	17

The state vector for cycle k is defined as

$$x(k) = \begin{bmatrix} d_1(k) & d_2(k) & d_3(k) & a_1(k) & a_2(k) & a_3(k) \end{bmatrix}^\top.$$

The reference signal, in this case the periodic timetable is defined as

$$r(k) = r(0) \otimes 30^{\otimes^k},$$

and the implicit SMPL model can be written as

$$\begin{aligned} x(k) = A_0(k,u(k)) \otimes x(k) \oplus A_1(k,u(k-1)) \otimes x(k-1) \oplus \\ A_{-1}(k,u(k)) \otimes x(k+1) \oplus r(k), \end{aligned}$$

with

$$\begin{aligned} A_0(k,u(k)) &= \begin{bmatrix} A_{0,a}(k,u(k)) & A_{0,b}(k,u(k)) \\ A_{0,c}(k,u(k)) & A_{0,d}(k,u(k)) \end{bmatrix} \\ A_1(k,u(k-1)) &= \begin{bmatrix} A_{1,a}(k,u(k-1)) & A_{1,b}(k,u(k-1)) \\ A_{1,c}(k,u(k-1)) & A_{1,d}(k,u(k-1)) \end{bmatrix} \\ A_{-1}(k,u(k)) &= \begin{bmatrix} A_{-1,a}(k,u(k)) & A_{-1,b}(k,u(k)) \\ A_{-1,c}(k,u(k)) & A_{-1,d}(k,u(k)) \end{bmatrix}, \end{aligned}$$

where $A_{0,a}(k,u(k))$ contains the headway times between the departure events of cycle k and is defined as

$$A_{0,a}(k,u(k)) = \begin{bmatrix} \varepsilon & 3 \otimes \overline{u_1(k)} & 3 \otimes \overline{u_3(k)} \\ 3 \otimes u_1(k) & \varepsilon & 3 \otimes \overline{u_2(k)} \\ 3 \otimes u_3(k) & 3 \otimes u_2(k) & \varepsilon \end{bmatrix}.$$

Since this simple example has no connections or trains traversing the track in opposing direction, we have: $A_{0,b}(k,u(k)) = \mathcal{E}$.

The running times are given by

$$A_{0,c}(k,u(k)) = \mathrm{diag}_\oplus(18,18,17).$$

The matrix $A_{0,d}(k,u(k))$ contains the headway times between the arrival events of cycle k, but since we assumed all headway times were 3 minutes, the matrix is the same as for the headway times between the departure events:

$$A_{0,d}(k,u(k)) = A_{0,a}(k,u(k)).$$

The matrix $A_1(k,u(k-1))$ only defines the headway times between trains of the current and previous cycle. The headway constraints are only found in $A_{1,a}(k,u(k-1))$ and $A_{1,d}(k,u(k-1))$; therefore $A_{1,b}(k,u(k-1)) = A_{1,c}(k,u(k-1)) = \mathcal{E}$. Since the headway times are all 3 minutes $A_{1,a}(k,u(k-1))$ is equal to $A_{1,d}(k,u(k-1))$ and defined as:

$$A_{1,a}(k,u(k-1)) = \begin{bmatrix} 3 \otimes u_7(k-1) & 3 \otimes u_5(k-1) & 3 \otimes u_4(k-1) \\ 3 \otimes u_{10}(k-1) & 3 \otimes u_8(k-1) & 3 \otimes u_6(k-1) \\ 3 \otimes u_{12}(k-1) & 3 \otimes u_{11}(k-1) & 3 \otimes u_9(k-1) \end{bmatrix},$$

and $u_4(k-1), \ldots, u_{12}(k-1)$ are max-plus binary variables used to determine the order of the trains.

Similarly $A_{-1}(k,u(k))$ only defines the headway times between trains of the current and next cycle; therefore $A_{-1,b}(k,u(k)) = A_{-1,c}(k,u(k)) = \mathcal{E}$ and since the headway times are all 3 minutes $A_{-1,a}(k,u(k))$ is equal to $A_{-1,d}(k,u(k))$ and defined as:

$$A_{-1,a}(k,u(k)) = \begin{bmatrix} 3 \otimes \overline{u_7(k)} & 3 \otimes \overline{u_{10}(k)} & 3 \otimes \overline{u_{12}(k)} \\ 3 \otimes \overline{u_5(k)} & 3 \otimes \overline{u_8(k)} & 3 \otimes \overline{u_{11}(k)} \\ 3 \otimes \overline{u_4(k)} & 3 \otimes \overline{u_6(k)} & 3 \otimes \overline{u_9(k)} \end{bmatrix}.$$

Now let us define the current cycle k as cycle 1, then $x(0) = r(0)$ and since there were no delays in the past no control actions had to be taken and therefore $u_i(k-1) = 0$, $\forall i$.

To keep the example small and illustrative a prediction and control horizon length of 16 minutes is chosen. Each time step is one minute: $\tau_{\mathrm{H}} = 1$ minute and time instant $\kappa = 0$ is $t(0) = 0$. We will look at time instant $t(29) = 29$ minutes, since at that time instant all events of cycle 0 have already occurred, and none of the events in cycle 1 have occurred yet. For this time instant $\chi(29)$ is given by:

$$\chi(29) = \begin{bmatrix} d_1(1) & d_2(1) & d_3(1) \end{bmatrix}^\top,$$

since only these event times of $x(1)$ are predicted and scheduled in the interval $[29, 45)$.

The max-plus binary variables that can be changed at this time are those that are in a constraint for which both continuous variables have their event times in $\chi(\kappa)$.

Only for the elements in $A_{0,a}(1,u(1))$ both event times are in $\chi(29)$, as a result the control variables in $\psi(29)$ are:

$$\psi(29) = \begin{bmatrix} u_1(1) & u_2(1) & u_3(1) \end{bmatrix}^\top.$$

Furthermore $u_i(1) = 0$ for $i = 4, \ldots, 12$ since they are associated to future events that are outside the prediction horizon and it is assumed no control actions are taken for those future events at time instant $t(\kappa)$. As a result $A_{-1}(k,u(k)) = \mathcal{E}$, simplifying the implicit SMPL model for the optimization problem at time instant $t(29)$ to:

$$\begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} = A_{0,a}(1,u(1)) \otimes \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} \oplus \begin{bmatrix} A_{1,a}(1,u(0)) & \mathcal{E} \end{bmatrix} \otimes x(0) \oplus \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix},$$

$$
\begin{aligned}
&= A_{0,a}(1,u(1)) \otimes \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} \oplus \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 5 \\ 10 \end{bmatrix} \oplus \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix} \\
&= A_{0,a}(1,u(1)) \otimes \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} \oplus \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix} \\
&= \begin{bmatrix} 3 \otimes \overline{u_1(1)} \otimes d_2(1) \oplus 3 \otimes \overline{u_3(1)} \otimes d_3(1) \oplus 30 \\ 3 \otimes u_1(k) \otimes d_1(1) \oplus 3 \otimes \overline{u_2(1)} \otimes d_3(1) \oplus 35 \\ 3 \otimes u_3(k) \otimes d_1(1) \oplus 3 \otimes u_2(1) \otimes d_2(1) \oplus 40 \end{bmatrix}.
\end{aligned}
$$

Now we convert the implicit SMPL model into a set of mixed-integer-linear constraints:

$$
\begin{aligned}
& d_1(1) = 3 \otimes \overline{u_1(1)} \otimes d_2(1) \oplus 3 \otimes \overline{u_3(1)} \otimes d_3(1) \oplus 30 \\
& \qquad \Rightarrow \\
& d_1(1) \geq (1 - \upsilon_1(1))\beta + d_2(1) + 3 \\
& d_1(1) \geq (1 - \upsilon_3(1))\beta + d_3(1) + 3 \\
& d_1(1) \geq 30 \\
\\
& d_2(1) = 3 \otimes u_1(1) \otimes d_1(1) \oplus 3 \otimes \overline{u_2(1)} \otimes d_3(1) \oplus 35 \\
& \qquad \Rightarrow \\
& d_2(1) \geq \upsilon_1(1)\beta + d_1(1) + 3 \\
& d_2(1) \geq (1 - \upsilon_2)(1)\beta + d_3(1) + 3 \\
& d_2(1) \geq 35 \\
\\
& d_3(1) = 3 \otimes u_3(1) \otimes d_1(1) \oplus 3 \otimes u_2(1) \otimes d_2(1) \oplus 40 \\
& \qquad \Rightarrow \\
& d_3(1) \geq \upsilon_3(1)\beta + d_1(1) + 3 \\
& d_3(1) \geq \upsilon_2(1)\beta + d_2(1) + 3 \\
& d_3(1) \geq 40.
\end{aligned}
$$

Now these constraints need to be transformed into the form of (4.6). First define

$$
z(29) = \begin{bmatrix} d_1(1) & d_2(1) & d_3(1) & \upsilon_1(1) & \upsilon_2(1) & \upsilon_3(1) \end{bmatrix}^\top,
$$

then rewrite the constraints such that all elements of $z(\kappa)$ are on one side of the inequality:

$$
\begin{aligned}
-z_4(29)\beta + z_2(29) - z_1(29) &\leq -\beta - 3 \\
-z_6(29)\beta + z_3(29) - z_1(29) &\leq -\beta - 3 \\
-z_1(29) &\leq -30 \\
z_4(29)\beta + z_1(29) - z_2(29) &\leq -3 \\
-z_5(29)\beta + z_3(29) - z_2(29) &\leq -3 - \beta \\
-z_2(29) &\leq -35
\end{aligned}
$$

$$\begin{aligned} z_6(29)\beta + z_1(29) - z_3(29) &\leq -3 \\ z_5(29)\beta + z_2(29) - z_3(29) &\leq -3 \\ -z_3(29) &\leq -40. \end{aligned}$$

The optimization problem at $t(29)$ is then given by:

$$\min_{z(29)} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} z_1(29) - 30 \\ z_2(29) - 35 \\ z_3(29) - 40 \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} z_1(29) \\ z_2(29) \\ z_3(29) \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} - 105$$

$$\text{s.t.} \quad \begin{bmatrix} -1 & 1 & 0 & -\beta & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & -\beta \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & \beta & 0 & 0 \\ 0 & -1 & 1 & 0 & -\beta & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & \beta \\ 0 & 1 & -1 & 0 & \beta & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(29) \\ z_2(29) \\ z_3(29) \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} \leq \begin{bmatrix} -\beta - 3 \\ -\beta - 3 \\ -30 \\ -3 \\ -3 - \beta \\ -35 \\ -3 \\ -3 \end{bmatrix}.$$

The implicit model can also be converted to its explicit form before define the optimization problem. Consider the reduced implicit SMPL model we derived earlier:

$$\begin{aligned} \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} &= A_{0,a}(1, u(1)) \otimes \begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} \oplus \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix} \\ &= \begin{bmatrix} 3 \otimes \overline{u_1(1)} \otimes d_2(1) \oplus 3 \otimes \overline{u_3(1)} \otimes d_3(1) \oplus 30 \\ 3 \otimes u_1(k) \otimes d_1(1) \oplus 3 \otimes \overline{u_2(1)} \otimes d_3(1) \oplus 35 \\ 3 \otimes u_3(k) \otimes d_1(1) \oplus 3 \otimes u_2(1) \otimes d_2(1) \oplus 40 \end{bmatrix}. \end{aligned}$$

The powers of $A_{0,a}(1, u(1))$ have already been calculated in Section 3.3.1. The matrix $A_{0,a}(1, u(1))^*$ is just the max-plus addition of these powers and the max-plus identity matrix E resulting in:

$$A_{0,a}(1, u(1))^* = \begin{bmatrix} As_1(1, u(1)) & As_2(1, u(1)) & As_3(1, u(1)) \end{bmatrix},$$

with

$$As_1(1, u(1)) = \begin{bmatrix} e \\ 3 \otimes u_1(1) \oplus 6 \otimes \overline{u_2(1)} \otimes u_3(1) \\ 3 \otimes u_3(1) \oplus 6 \otimes u_1(1) \otimes u_2(1) \end{bmatrix}$$

$$As_2(1, u(1)) = \begin{bmatrix} 3 \otimes \overline{u_1(1)} \oplus 6 \otimes u_2(1) \otimes \overline{u_3(1)} \\ e \\ 3 \otimes u_2(1) \oplus 6 \otimes \overline{u_1(1)} \otimes u_3(1) \end{bmatrix}$$

$$As_3(1,u(1)) = \begin{bmatrix} 3 \otimes \overline{u_3(1)} \oplus 6 \otimes \overline{u_1(1)} \otimes \overline{u_2(1)} \\ 3 \otimes \overline{u_2(1)} \oplus 6 \otimes u_1(1) \otimes \overline{u_3(1)} \\ e \end{bmatrix},$$

with the following constraints ensuring infeasible combinations of max-plus binary variables are not chosen:

$$\overline{u_{2,1}(k)} \otimes \overline{u_{3,2}(k)} \otimes u_{3,1}(k) \leq \varepsilon$$
$$u_{2,1}(k) \otimes u_{3,2}(k) \otimes \overline{u_{3,1}(k)} \leq \varepsilon.$$

This results in the following feasible explicit SMPL:

$$\begin{bmatrix} d_1(1) \\ d_2(1) \\ d_3(1) \end{bmatrix} = A_{0,a}(1,u(1))^* \otimes \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix}.$$

This should then be rewritten into a set of mixed-integer-linear constraints and rewritten such that all elements of $z(\kappa)$ are on one side of the inequality as we have shown before.

The resulting optimization problem at $t(29)$ is then given by:

$$\min_{z(29)} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} z_1(29) - 30 \\ z_2(29) - 35 \\ z_3(29) - 40 \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} z_1(29) \\ z_2(29) \\ z_3(29) \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} - 105$$

$$\text{s.t.} \quad \begin{bmatrix} -1 & 0 & 0 & -\beta & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -\beta \\ -1 & 0 & 0 & 0 & \beta & -\beta \\ -1 & 0 & 0 & -\beta & -\beta & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & \beta & 0 & 0 \\ 0 & -1 & 0 & 0 & -\beta & 0 \\ 0 & -1 & 0 & 0 & -\beta & \beta \\ 0 & -1 & 0 & \beta & 0 & -\beta \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & \beta \\ 0 & 0 & -1 & 0 & \beta & 0 \\ 0 & 0 & -1 & \beta & \beta & 0 \\ 0 & 0 & -1 & -\beta & 0 & \beta \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} z_1(29) \\ z_2(29) \\ z_3(29) \\ z_4(29) \\ z_5(29) \\ z_6(29) \end{bmatrix} \leq \begin{bmatrix} -\beta - 38 \\ -\beta - 43 \\ -41 - \beta \\ -46 - 2\beta \\ -30 \\ -33 \\ -\beta - 43 \\ -36 - \beta \\ -46 - \beta \\ -35 \\ -33 \\ -38 \\ -36 \\ -41 - \beta \\ -40 \\ 1 \\ 0 \end{bmatrix}.$$

4.2.7 Explicit SMPL and the cost function

For the explicit model the choice of the cost function affects the number of constraints in (4.6) since by construction the state vector $\breve{x}(k)$ only depends on the max-plus binary

variables and the state vectors of previous cycles $\breve{x}(k-\mu)$ for $\mu \in \{1,\dots,\mu_{\max}\}$ and not on $\breve{x}(k)$ itself. As a result if the cost function only weighs a subset of the event times with nonzero weights, such as the arrival times and not the departure times, then the part of the explicit SMPL model describing the departure times can be removed as shown here.

To show why part of the explicit SMPL model can be removed we will look at the structure of the explicit switching max-plus-linear model as described in (3.50) and repeated here for convenience:

$$\breve{x}(k) = \breve{A}_0^{*,\text{feas}} \otimes \left(\breve{r}(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \breve{A}_\mu \otimes x(k-\mu) \right).$$

We expand it by splitting the state vector into the arrival and departure delays:

$$\begin{aligned}
\breve{x}(k) &= \breve{A}_0^{*,\text{feas}} \otimes \breve{r}(k) \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \breve{A}_{\text{exp},\mu} \otimes x(k-\mu) \\
\begin{bmatrix} \breve{d}(k) \\ \breve{a}(k) \end{bmatrix} &= \begin{bmatrix} \breve{A}_{0,\text{a}}^{*,\text{feas}} & \breve{A}_{0,\text{b}}^{*,\text{feas}} \\ \breve{A}_{0,\text{c}}^{*,\text{feas}} & \breve{A}_{0,\text{d}}^{*,\text{feas}} \end{bmatrix} \otimes \begin{bmatrix} \breve{r}_\text{d}(k) \\ \breve{r}_\text{a}(k) \end{bmatrix} \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \begin{bmatrix} \breve{A}_{\text{exp},\mu,\text{a}} & \breve{A}_{\text{exp},\mu,\text{b}} \\ \breve{A}_{\text{exp},\mu,\text{c}} & \breve{A}_{\text{exp},\mu,\text{d}} \end{bmatrix} \begin{bmatrix} \breve{d}(k-\mu) \\ \breve{a}(k-\mu) \end{bmatrix},
\end{aligned}$$

where $\breve{A}_{\text{exp},\mu} = \breve{A}_0^{*,\text{feas}} \otimes \breve{A}_\mu$, and

$$\begin{aligned}
\breve{A}_0^{*,\text{feas}} &= \begin{bmatrix} \breve{A}_{0,\text{a}}^{*,\text{feas}} & \breve{A}_{0,\text{b}}^{*,\text{feas}} \\ \breve{A}_{0,\text{c}}^{*,\text{feas}} & \breve{A}_{0,\text{d}}^{*,\text{feas}} \end{bmatrix} \\
\breve{A}_{\text{exp},\mu} &= \begin{bmatrix} \breve{A}_{\text{exp},\mu,\text{a}} & \breve{A}_{\text{exp},\mu,\text{b}} \\ \breve{A}_{\text{exp},\mu,\text{c}} & \breve{A}_{\text{exp},\mu,\text{d}} \end{bmatrix}.
\end{aligned}$$

This can be split into two equations, one to calculate the arrivals and one to calculate the departures:

$$\breve{d}(k) = \begin{bmatrix} \breve{A}_{0,\text{a}}^{*,\text{feas}} & \breve{A}_{0,\text{b}}^{*,\text{feas}} \end{bmatrix} \otimes \begin{bmatrix} \breve{r}_\text{d}(k) \\ \breve{r}_\text{a}(k) \end{bmatrix} \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \begin{bmatrix} \breve{A}_{\text{exp},\mu,\text{a}} & \breve{A}_{\text{exp},\mu,\text{b}} \end{bmatrix} \begin{bmatrix} \breve{d}(k-\mu) \\ \breve{a}(k-\mu) \end{bmatrix} \tag{4.7}$$

$$\breve{a}(k) = \begin{bmatrix} \breve{A}_{0,\text{c}}^{*,\text{feas}} & \breve{A}_{0,\text{d}}^{*,\text{feas}} \end{bmatrix} \otimes \begin{bmatrix} \breve{r}_\text{d}(k) \\ \breve{r}_\text{a}(k) \end{bmatrix} \oplus \bigoplus_{\mu=1}^{\mu_{\max}} \begin{bmatrix} \breve{A}_{\text{exp},\mu,\text{c}} & \breve{A}_{\text{exp},\mu,\text{d}} \end{bmatrix} \begin{bmatrix} \breve{d}(k-\mu) \\ \breve{a}(k-\mu) \end{bmatrix}. \tag{4.8}$$

In these equations the arrival delays only depend on the arrival and departure delays of the previous cycles and not on the departure delays of the current cycle. If we only want to determine the arrival delays, for example in the case we are only interested in minimizing the sum of arrival delays, then we do not need to calculate the departure delays. The same is true for any other subset of event times of $\breve{x}(k)$.

4.3 Distributed model predictive control

For very large and complex systems the optimization problem resulting from a centralized MPC approach may not be solvable within the time available, or the solution found may be far from optimal. A solution for this problem can be distributed model predictive control

(DMPC) [9, 29, 31, 76]. Most DMPC approaches have been developed for continuous or discrete-time systems [69]. Since our model is a discrete-event model, these DMPC approaches cannot be directly applied to our problem. Instead, we develop our own DMPC methods.

In DMPC the system is partitioned into a number of subsystems and each subsystem is controlled by its own model predictive controller. The controllers determine the optimal control inputs for the subsystems while interacting with the controllers of the other subproblems to ensure global feasibility and good global performance. Each subsystem only contains a part of the dynamics of the whole system and the controller only controls that part of the system. The controller may model the effects of its control actions on the rest of the network, but for the control actions in the rest of the network the controller depends on the other subsystems and their controllers. In the case of railway traffic management each subsystem consists of a subset of all stations and tracks and the trains on that part of the network and the controller of the subsystem can only control that part of the network. In that part of the network the controller can take any dispatching action. In the rest of the network the other controllers decide the schedule.

The advantage of partitioning the system into subsystems is that the optimization problem that must be solved to find the control inputs for the subsystems can in general be solved much faster since it is smaller and less complex than the optimization problem used to determine the control inputs for the centralized MPC method. The downside is that the control inputs found may be suboptimal for the complete system.

4.3.1 Model-based partitioning

In this section we propose several DMPC methods to find the dispatching actions for the whole network.

Before the DMPC methods are explained we will first reorder the rows and columns of the constraint matrix A and reorder the variables in vector z such that the structure of A gets as close as possible to a block-diagonal structure. The number of blocks depends on the choice of the number of subsystems which is n_{sub}. The number of subsystems can be chosen by the designer, but there is limit to the number of subsystems which depends on the system itself.

This reordering is based on the following goals:

- The constraints that cannot be placed in the block diagonal structure should only depend on continuous variables.
- Each diagonal block has its own unique set of binary and continuous variables it depends on, there is no overlap between those sets, and the union of the sets contains all the binary and continuous variables.
- The number of constraints that cannot be placed in the block diagonal structure should be minimized.
- The size of the blocks should be of the same magnitude.

- The difference in the number of binary variables and continuous variables each block depends on should be minimized.

If running time constraints are outside the block diagonal structure the headway constraints between the arrival and departures of the trains, which depend on the same binary variables, will be in separate blocks, and as a result two blocks depend on the same binary variables, which means the second goal would not be achieved. Therefore running time constraints should be in the block diagonal structure to ensure the first two goals. Furthermore headway, separation, and breakable connection constraints all depend on binary variables, and therefore should not be outside the block-diagonal structure to ensure the first two goals. Only continuity and unbreakable connection constraints remain. By only allowing continuity and unbreakable connection constraints to be outside the block-diagonal structure, the first two goals can be achieved.

If we look at the railway model this means that the constraints describing the trains traversing a track, the headway, and separation constraints between the trains on that track should all be in the same block.

The steps to reorder the constraint matrix such that it has n_{sub} diagonal blocks are as follows:

1) Group the variables and constraints per track, resulting in n_{T} sets of constraints and variables, where n_{T} is the number of tracks in the model as defined in Chapter 3.

2) Merge all sets that are connected via a breakable connection constraint, resulting in n_{T2} remaining sets.

3) Solve a mixed integer quadratic programming (MIQP) problem that minimizes the sum of the maximum difference in the number of constraints of the n_{sub} subproblems and minimizes the number of constraints connecting the n_{sub} subproblems. The values of the binary variables determine for each of the n_{T2} sets to which of the n_{sub} subproblem they are assigned to.

The MIQP problem can be set up as follows:

- Define n_{sub} continuous variables, denoted by S_i for $i = 1, \ldots, n_{\text{sub}}$ that represent the number of constraints each subproblem has.

- Define one continuous variable $S_{\max}$ that is the maximum difference between the values S_i for $i = 1, \ldots, n_{\text{sub}}$.

- For each constraint set CON_j, $j = 1, \ldots, n_{\text{T2}}$ define n_{sub} binary variables denoted by $\upsilon_{j,i}$, for $i = 1, \ldots, n_{\text{sub}}$. If binary variable $\upsilon_{j,i} = 1$ then set CON_j is part of block i of the reordered constraint matrix.

- Define the cost function as

$$J = \varrho S_{\max} - \sum_{j=1}^{n_{\text{T2}}} \sum_{k=1}^{n_{\text{T2}}} \sum_{i=1}^{n_{\text{sub}}} \upsilon_{j,i} Q_{j,k} \upsilon_{k,i},$$

where ϱ is a tuning parameter that determines the importance of minimizing $S_{\max}$. The element $Q_{j,k}$ has the value equal to the number of constraints connecting set CON_k to set CON_j. For each combination of two sets CON_j and CON_k the cost function reduces in value equal to $Q_{j,k}$ if and only if both sets are in the same subproblem. This ensures that the optimization problem will minimize the number of constraints connecting the resulting n_{sub} blocks of the constraint matrix.

- In order for $S_{\max}$ to be equal to the maximum difference between the values S_i for $i = 1, \ldots, n_{\mathrm{sub}}$ the following set of constraints is used:

$$S_{\max} \geq S_i - S_j \quad \text{for } i \in \{1, \ldots, n_{\mathrm{sub}}\},\ j \in \{1, \ldots, n_{\mathrm{sub}}\} \setminus \{i\}.$$

- To ensure each constraint set is assigned to exactly one block of the constraint matrix the following set of constraints is used:

$$\sum_{i=1}^{n_{\mathrm{sub}}} \upsilon_{j,i} = 1 \quad \text{for } j = 1, \ldots, n_{\mathrm{T2}}.$$

- The number of constraints each block has is determined by the following constraints:

$$S_i \geq \sum_{j=1}^{n_{\mathrm{T2}}} \upsilon_{j,i} s_j \quad \text{for } i \in \{1, \ldots, n_{\mathrm{sub}}\},$$

where s_j is the number of constraints of constraint set CON_j.

Once the constraint matrix has been reordered the optimization problem at time instant $t(\kappa)$ of the MPC problem can be written as

$$\min_{z(\kappa)} \begin{bmatrix} \tilde{c}_1^\top(\kappa) & \tilde{c}_2^\top(\kappa) & \ldots & \tilde{c}_{n_{\mathrm{sub}}}^\top(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1^\top(\kappa) & \tilde{z}_2^\top(\kappa) & \ldots & \tilde{z}_{n_{\mathrm{sub}}}^\top(\kappa) \end{bmatrix}^\top \tag{4.9}$$

$$\text{s.t.} \begin{bmatrix} A_{1,1}(\kappa) & A_{1,2}(\kappa) & \ldots & A_{1,n_{\mathrm{sub}}}(\kappa) \\ A_{2,1}(\kappa) & \ddots & & A_{2,n_{\mathrm{sub}}}(\kappa) \\ \vdots & & \ddots & \vdots \\ A_{n_{\mathrm{sub}},1}(\kappa) & A_{n_{\mathrm{sub}},2}(\kappa) & \ldots & A_{n_{\mathrm{sub}},n_{\mathrm{sub}}}(\kappa) \end{bmatrix} \begin{bmatrix} \tilde{z}_1(\kappa) \\ \tilde{z}_2(\kappa) \\ \vdots \\ \tilde{z}_{n_{\mathrm{sub}}}(\kappa) \end{bmatrix} \leq \begin{bmatrix} \tilde{b}_1(\kappa) \\ \tilde{b}_2(\kappa) \\ \vdots \\ \tilde{b}_{n_{\mathrm{sub}}}(\kappa) \end{bmatrix}, \tag{4.10}$$

where $\tilde{c}_i(\kappa), \tilde{z}_i(\kappa), \tilde{b}_i(\kappa)$ for $i \in \{1, \ldots, n_{\mathrm{sub}}\}$ are vectors of appropriate size, $A_{i,j}(\kappa)$ for $i \in \{1, \ldots, n_{\mathrm{sub}}\}, j \in \{1, \ldots, n_{\mathrm{sub}}\}$ are matrices of appropriate size, and

$$\begin{aligned} \tilde{z}_i(\kappa) &= \begin{bmatrix} \tilde{\chi}_i^\top(\kappa) & \tilde{\upsilon}_i^\top(\kappa) \end{bmatrix}^\top \\ A_{i,i}(\kappa) &= \begin{bmatrix} A_{i,i,x}(\kappa) & A_{i,i,\upsilon}(\kappa) \end{bmatrix} \\ A_{i,j}(\kappa) &= \begin{bmatrix} A_{i,j,x}(\kappa) & \mathbf{0}(\kappa) \end{bmatrix}, \end{aligned}$$

for $i \in \{1, \ldots, n_{\mathrm{sub}}\}$ and $j \in \{1, \ldots, n_{\mathrm{sub}}\} \setminus \{i\}$, and where $\tilde{\chi}_i(\kappa)$ and $\tilde{\upsilon}_i(\kappa)$ are vectors of appropriate size. The matrices $A_{i,i}(\kappa)$ are split up into a part $A_{i,i,x}(\kappa)$ that is multiplied by $\tilde{\chi}_i(\kappa)$, and a part $A_{i,i,\upsilon}(\kappa)$ that is multiplied by $\tilde{\upsilon}_i(\kappa)$ in (4.10). The matrices $A_{i,j}(\kappa)$ are also split up into two parts. One part $A_{i,j,x}(\kappa)$ is multiplied by $\tilde{\chi}_j(\kappa)$ in (4.10). The part that is multiplied by $\tilde{\upsilon}_j(\kappa)$ in (4.10) is guaranteed to be a zero matrix.

4.3.2 Distributed method 1

Based on the reordered constraint matrix two methods for solving the problem in a distributed manner are developed. The resulting structure of matrix shows that there are only a few constraints connecting each of the blocks, so there is very little interaction between the variables of each block. Therefore it makes sense to develop DMPC methods based on this block structure. For the first method each subproblem optimizes the centralized cost function while considering all constraints of the centralized problem, but for subproblem i at time instant $t(\kappa)$ only $\chi(\kappa)$ and $\tilde{v}_i(\kappa)$ can be used to optimize the cost function. By using the same cost function as the centralized problem, each subproblem considers the effects of its dispatching actions on the entire network. By limiting the binary variables that can be changed to $v_i(\kappa)$ the subproblem is computationally less complex to solve than the centralized problem. As a result the MILP problem can be solved much faster. The subproblems are then solved in sequence. When solving a subproblem the binary variables of the other subproblems are fixed to the value found when their respective subproblems were solved previously. Since all subproblems share the same cost function; the cost function of the centralized problem, every time a subproblem is solved the centralized cost function is reduced. This continuous until none of the subproblems can improve the cost function anymore. A feasible initial solution can always be found as follows. The simplest feasible solution is the solution when no dispatching action is taken; more specifically, all binary variables will be zero, and the trains would just continue to drive as regular and no order would be changed or connections broken.

To summarize the method:

Each subproblem can be written as[2]:

$$\begin{aligned} \min_{\chi(\kappa),\tilde{v}_i(\kappa)} \quad & c^\top(\kappa)\, z(\kappa) \\ \text{s.t.} \quad & A(\kappa)\, z(\kappa) \leq b(\kappa). \end{aligned}$$

The steps to determine the solution are:

1) Define an initial estimate for the solution of the centralized MPC $z(\kappa)$ denoted by $\hat{z}(\kappa)$.

2) Use $\hat{z}(\kappa)$ as an initial solution for subproblem i:

$$\begin{aligned} \min_{\chi(\kappa),\tilde{v}_i(\kappa)} \quad & c^\top(\kappa)\, z(\kappa) \\ \text{s.t.} \quad & A(\kappa)\, z(\kappa) \leq b(\kappa). \end{aligned}$$

   and solve it. Denote the solution as $\hat{z}^i(\kappa)$. The initial solution is needed to update the values of the binary variables that have been determined by the other subproblems and cannot be changed by subproblem i.

3) Update the estimate of the solution of the centralized MPC: $\hat{z}(\kappa) = \hat{z}^i(\kappa)$

[2]Recall that $z(\kappa)$ is split up into n_{sub} sub-vectors $\tilde{z}_i(\kappa)$, and $\tilde{v}_i(\kappa)$ are the binary variables in $\tilde{z}_i(\kappa)$ and are part of $z(\kappa)$.

4) Repeat steps 2 and 3 for the other subproblems.

5) Repeat steps 2, 3, and 4 until $\hat{z}(\kappa)$ no longer changes.

The initial estimate can be determined by a heuristic method such as first-come, first-served, or can be the result when no dispatching actions are taken, which can be determined very fast.

Since a feasible initial estimate can always be found, and all subproblems consider the centralized problem, but can only change a limited number of binary variables, a feasible solution for the centralized problem will always be found in step 2. Furthermore every feasible solution found can and will be used as a starting solution for the next subproblem. Therefore, every subproblem either improves the found solution or cannot find a better solution than the current solution. If no better solution than the current solution is found, the current solution is used for the next subproblem. Once no subproblem can improve the solution the algorithm stops. We call this method "DMPC method 1".

4.3.3 Distributed method 2

Since the number of constraints of each subproblem for DMPC method 1 is equal to the number of constraints for the central MPC problem, for very large systems the speed up in computation time may still not enough. For very large systems increasing the number of subproblem is not a solution either, since the number of constraints remain the same. Therefore another DMPC approach is proposed.

For the second DMPC method each subproblem only considers a part of the MPC problem in (4.10). It only optimizes the continuous and binary variables of one of the diagonal blocks. Each subproblem can then be written as:

$$\min_{\tilde{z}_i(\kappa)} \tilde{c}_i^\top(\kappa)\,\tilde{z}_i(\kappa) \tag{4.11}$$

$$\text{s.t. } A_{i,i}(\kappa)\tilde{z}_i(\kappa) \leq b_i(\kappa) - \sum_{j\in\{1,\ldots,n_{\text{sub}}\}\setminus\{i\}} A_{i,j}(\kappa)\tilde{z}_j(\kappa), \tag{4.12}$$

where $\tilde{z}_j(\kappa)$, for $j \in \{1,\ldots,n_{\text{sub}}\}\setminus\{i\}$ is determined by the other local model predictive controllers.

The steps to determine a feasible centralized solution are:

1) Define an initial estimate for the solution of the centralized MPC $z(\kappa)$ denoted by $\hat{z}(\kappa)$, define the iteration counter $l = 0$, and define $\hat{\tilde{z}}_i^l(\kappa) = \hat{z}_i(\kappa)$ for $i = 1,\ldots,n_{\text{sub}}$.

2) Increase the iteration counter by one: $l = l+1$. For subproblem i, denoted by (4.11) and (4.12), assume $\tilde{z}_j(\kappa)$ for $j \in \{1,\ldots,n_{\text{sub}}\}\setminus\{i\}$ is known and equal to $\hat{\tilde{z}}_j(\kappa)$ and solve subproblem i for $i = 1$. Denote the solution as $\hat{\tilde{z}}_i^l(\kappa)$.

3) Update the estimate $\hat{\tilde{z}}_i(\kappa) = \hat{\tilde{z}}_i^l(\kappa)$.

4) Repeat steps 2 and 3 for $i = 2,\ldots,n_{\text{sub}}$.

5) Repeat steps 2, 3, and 4 until $|\hat{\tilde{z}}_i^l(\kappa) - \hat{\tilde{z}}_i^{l-1}(\kappa)| < \Delta$ for $i = 1, \ldots, n_{\text{sub}}$, where Δ is a very small value. If there is no convergence within $\text{iter}_{\max}$ iterations then lock the binary variables $\tilde{v}(\kappa)$ to their last determined values and determine the corresponding continuous variables $\tilde{\chi}(\kappa)$.

A feasible solution can always be found in step 2, since taking no dispatching actions will always result in a feasible solution; the delays however may be much higher in that case compared to the optimal solution.

The advantage of this method is that the subproblems are even smaller and can be solved faster. However since all subproblems only consider a part of the network, the subproblems have no way of taking into account the effects of their control actions on the other subproblems. It is therefore likely that the solutions found will be worse than those found with method 1. We will simply call this "DMPC method 2".

4.3.4 Adjusting cost functions

For the second DMPC method in this section the local model predictive controllers do not consider the delay propagation to the other parts of the network and as a result the train orders at the border of their area of control tend to be suboptimal for the total network, since a large part of the effects of the order changes are only noticed in the next area. To reduce the delay propagation from one area controlled by a local model predictive controller to another area we will improve the second method by adjusting the weights of the events of the trains leaving to another area. The weights are adjusted only at the start of the optimization at each time instant. The weights that need to be adjusted can be found easily. For the local model predictive controller i the indices of the weights of the MILP problem in $\tilde{c}_i(\kappa)$ that we want to adjust can be determined by finding the variables of MILP problem i that are connected to MILP problem j for $j \in \{1, \ldots, n_{\text{sub}}\} \setminus \{i\}$ through a constraint.

If we look at how MILP problem i is defined in (4.11) and (4.12) it is clear that the indices of the variables of MILP problem i on which MILP problem j depends coincide with the indices of the non-zero columns in $A_{j,i}(\kappa)$. The indices of $\tilde{c}_i(\kappa)$ that need to be adjusted therefore coincide with the indices of the non-zero columns of $A_{j,i}(\kappa)$ for $i = 1, \ldots, n_{\text{sub}}$ and $j \in \{1, \ldots, n_{\text{sub}}\} \setminus \{i\}$. Denote these weights by $\tilde{c}_i^{\text{out}}(\kappa)$ and denote the corresponding variables as $\tilde{z}_i^{\text{out}}(\kappa)$.

By increasing the weights $\tilde{c}_i^{\text{out}}(\kappa)$ the local controllers are forced to focus on reducing the delays of the corresponding events, and by doing so possibly reduce the delay propagation to the other areas, at the cost of the total delay in the local area. There are many possible choices for the values of the weights $\tilde{c}_i^{\text{out}}(\kappa)$. In this chapter we will consider two different choices:

- Increase the values of weights $\tilde{c}_i^{\text{out}}(\kappa)$ with the same factor.
- Increase the values of weights $\tilde{c}_i^{\text{out}}(\kappa)$ based on the number of continuous variables the corresponding trains have in the other MILP problems.

The first option is really simple: just choose a factor with which to increase all weights

$\tilde{c}_i^{\text{out}}(\kappa)$. Our choice is to double the weights. The reason for this is that every continuous variable corresponding to these weights is connected to at least one continuous variable in one of the other MILP problems, and the dependency between the delays of these two continuous variables is very high. We call this method "DMPC method 3".

For the second option we need to determine for every train leaving the area which events are used to model this train in the other areas. This corresponds to the continuous variables that are connected through continuity and running time constraints to the variables $\tilde{z}_i^{\text{out}}(\kappa)$. Denote the set containing these continuous variables by $\mathcal{Z}_i^{\text{out}}(\kappa)$. The weights $\tilde{c}_i^{\text{out}}(\kappa)$ are then set to:

$$\tilde{c}_i^{\text{out}}(\kappa) = \tilde{c}_i^{\text{out}}(\kappa) + \sum_{\tilde{z}_j(\kappa) \in \mathcal{Z}_i^{\text{out}}(\kappa)} \tilde{c}_j(\kappa), \tag{4.13}$$

which is the sum of the weights of the standard cost function of the continuous variables in $\mathcal{Z}_i^{\text{out}}(\kappa)$ plus the weight on $\tilde{z}_i^{\text{out}}(\kappa)$. We call this method "DMPC method 4". In the next section we will show what the effects of these changes to the cost function are on the quality of the solution found with the DMPC methods and what influence they have on the computation time.

4.4 Case studies: Implicit versus explicit MPC

In this section the effects of the reduction method in combination with the explicit model on the solution time of the rescheduling problem will be evaluated. Furthermore, the effects of different objective functions on the solution found and computation time will be evaluated via a case study. In this case study we will look at a single step of the model predictive controller.

The case study will be based on the Dutch Railway network and the timetable of the year 2004. For the running times a 4% buffer time is assumed. For the dwell times a 2% buffer time is assumed. At end stations a 10 minute buffer time is assumed. The model is simplified because only intercity and interregional trains are considered, no local trains are considered. he headway times between the trains are assumed to be 3 minutes. Furthermore, only stations and junctions are considered where the trains can be rescheduled. Arrivals and departures at stations where trains can only stop and not overtake are not explicitly modeled. The only dispatching actions in this case study involve the reordering of trains. The timetable period is one hour and we test the MPC methods for a prediction horizon of 60 and 120 minutes. The resulting model consists of 40 stations with 109 tracks connecting the stations. Per hour 164 trains are considered.

Delays in the network can be divided into two types: "unavoidable" and "avoidable" delays. For each train the unavoidable delays are the delays that cannot be avoided. They are caused by increased process times of that train or because of trains that hinder it and cannot be avoided by rescheduling. The avoidable delays of a train are the delays that can be avoided by giving that train the highest priority and letting it leave as soon as possible, ignoring all other trains. For the entire network none of the unavoidable delays can be recovered with the use of dispatching actions. Only the avoidable delays of some

trains can be recovered with rescheduling. Not all avoidable delays can be recovered since some rescheduling actions may reduce the avoidable delays of one train, but increase the avoidable delays of another train. The goal is to minimize the sum of avoidable delays in the entire network.

To test the effectiveness of the rescheduling method and the computation time needed for the implicit and reduced explicit model we have built a set of 500 delay scenarios. We consider the railway traffic for two hours. In the first hour for each scenario 20% of the trains are randomly selected and given a random unavoidable delay by increasing the running time of those trains. The value of the delay is determined by a Weibull distribution [92] with scale parameter 6 and shape parameter 0.8. The shape parameter is based on the data on arrival and departure delays from [94]. The scale parameter is chosen such that the average delay is around 5 to 6 minutes. The maximum delay is set to 12 minutes, meaning we cut off the Weibull distribution at 12 minutes. Any delay above 12 minutes is set to 12 minutes. The resulting delays have a mean of 5.2 minutes and a standard deviation of 4.3 minutes. The model predictive controller then optimizes the dispatching actions for the next hour. In this hour no new unavoidable delays are introduced. The only delays present in this hour are the delays that propagated from the unavoidable delays in the previous period. The same is also done with a model predictive controller that optimizes the dispatching actions for the next two hours. Determining the optimal dispatching actions can be done by solving an MILP problem.

The value for the maximum delay used in the reduction method is set to 15 minutes, since many trains in the busiest parts of the Dutch railway network drive once every 15 minutes and for larger delays it is more likely that these trains are canceled.

All calculations are done on an AMD Phenom II X4 960T at 3GHz with 16GB of memory, running 64bit Windows 7. The model is built in MATLAB and all solvers are called using the mex interface of MATLAB. The solvers used are GLPK 4.46 [35], Gurobi 5.60 [40], and TOMLAB/CPLEX 12.5 [81].

4.4.1 Case study 1: Minimization of the sum of delays

In this section we compare the performance of the MPC problems based on the implicit and reduced explicit models described in Chapter 3. The cost function we will use is the sum of all delays: $c(\kappa)^\top z(\kappa) = [\mathbf{1}^\top(\chi(\kappa)) \quad 0.0001 \times \mathbf{1}^\top(\upsilon(\kappa))]\,[x^\top(\kappa) \quad \upsilon^\top(\kappa)]^\top$, where $\mathbf{1}(\chi(\kappa))$ and $\mathbf{1}(\upsilon(\kappa))$ are column vectors of appropriate size containing only ones. The implicit model is converted to the explicit model and reduced using the reduction method of Section 3.5 with an assumed maximum delay of 15 minutes.

First we will illustrate the structure of the constraint matrices of the optimization problems. The size of the constraint matrices vary from scenario to scenario because the exact set of trains and dispatching actions that are considered depends on the delay scenario. In general, the structure of the matrices remains the same for the different delay scenarios, just the number of constraints varies. The constraints in the implicit MILP problem all have one or two continuous variables. The constraints in the explicit MILP problem all have one continuous variable, except for the constraints used to ensure that certain combinations of control variables are not chosen, such as the combinations that

result in an infeasible train order and the combinations of control variables removed by the reduction method. Those constraints have no continuous variables in them. For an example see Section 4.2.6. The general structure of the constraint matrices is shown in Figures 4.1 and 4.2 for the implicit and reduced explicit MILP problem.

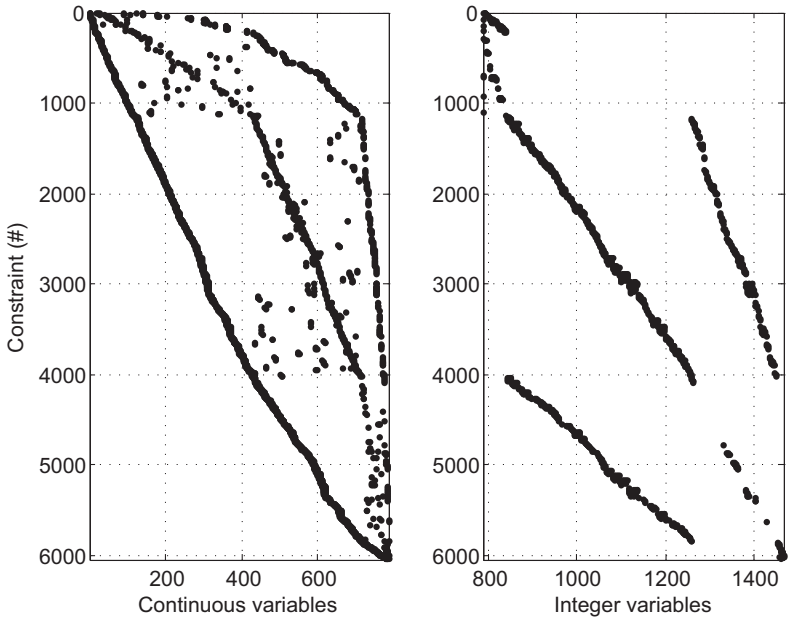

Figure 4.1: Structure of the constraint matrix of the implicit MILP problem.

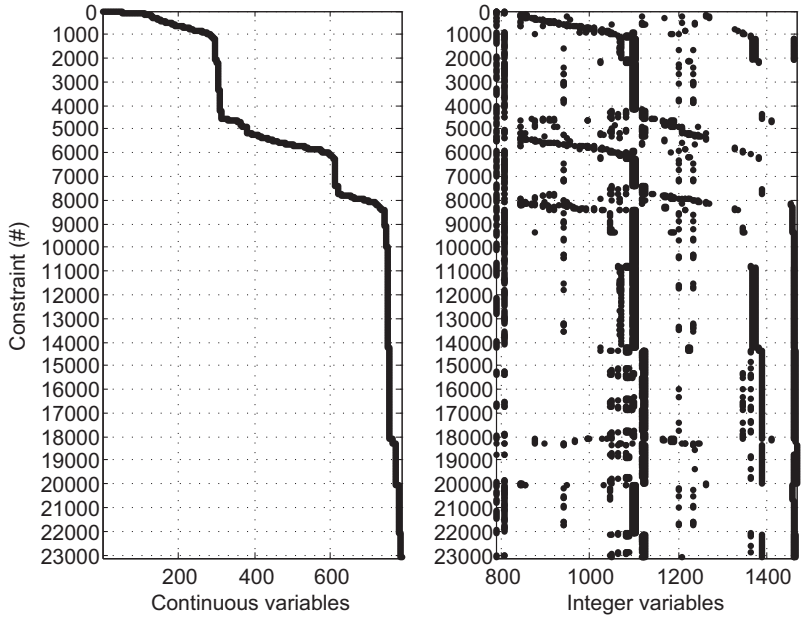

Figure 4.2: Structure of the constraint matrix of the explicit MILP problem.

Case study 1.1: A prediction horizon of one hour

The first case study consists of 500 scenarios where in each scenario two hours of the railway traffic is modeled. In the first hour the delays are built up and then for the second hour the model predictive controller is used to minimize the sum of all delays. The conversion into and reduction of the explicit model is done off-line and takes around 40 seconds to complete for the prediction horizon of one hour.

The average, minimum, and maximum number of continuous variables, binary variables, and constraints for the implicit and explicit model for a prediction horizon of one hour are given in Table 4.2.

Table 4.2: Number of constraints, continuous and binary variables of the MILP problems for a 1 hour prediction horizon.

	Average	min	max
Continuous variables	774	759	790
Binary variables	671	633	711
Constraints-Implicit	5946	5794	6118
Constraints-Explicit	3591	3109	4325

For the 500 scenarios we will first look at how much the avoidable delays are reduced by applying control. First the delays are calculated when no reordering is applied. The resulting delay is considered the nominal case. The unavoidable delays are calculated per train by allowing it to run free after the delays have occurred (all other trains are ignored). The difference between the nominal case and the unavoidable delays we consider the avoidable delays.

This is shown in Figure 4.3. On average the reduction in avoidable delays is 34.17%. In three scenarios there was no reduction and no increase in avoidable delays.

To compare the distribution of delays we have taken all delays over the entire one hour prediction horizon of the 500 scenarios together. We have only considered the events that were delayed in the controller or uncontrolled case, or delayed in both cases. Events that were not delayed in either the uncontrolled or controlled case are not considered. By doing so the cumulative distribution of the delays starts at 0 for a 0 minute delay in the uncontrolled case. For these events the distributions are shown in Figure 4.4 for the uncontrolled and controlled case. By comparing the two distributions, it is clear that in the controlled case several events are no longer delayed, since the probability of a 0 minute delay is about 7%, meaning that 7% of the events that were delayed in the nominal case are no longer delayed in the controlled case. From the comparison of the distributions in Figure 4.4, it is also clear that in the controlled case there are more short delays and less longer delays. The longest delays are at most three minutes larger for the controlled case. But there are very few of those delays.

Next we will look at the computation time of the solution of the MILP problems with the use of the solvers GLPK 4.46, CPLEX 12.5, and Gurobi 5.60 for the implicit

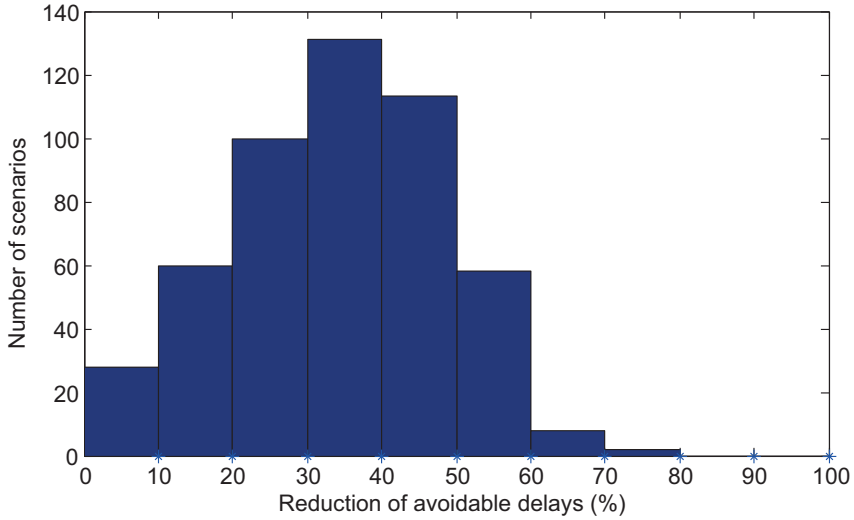

Figure 4.3: Histogram of the reduction of avoidable delays for the 500 scenarios for the one hour prediction horizon.

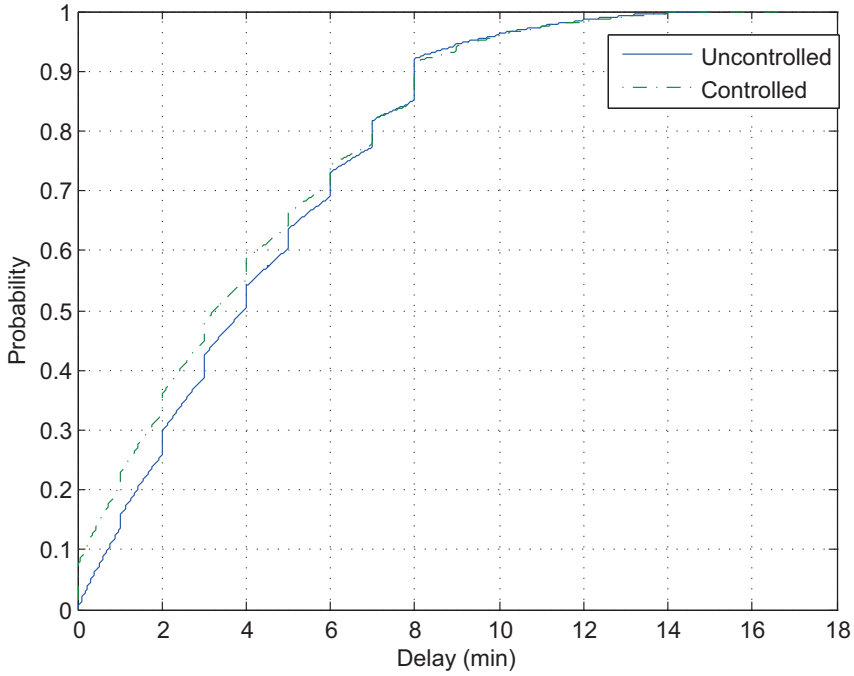

Figure 4.4: Cumulative distribution of the delays for the controlled and uncontrolled case for the one hour prediction horizon over all scenarios.

and explicit models. All calculations are done on an AMD Phenom II X4 960T at 3GHz with 16GB of memory, running 64bit Windows 7. The model is built in MATLAB and all solvers are called using the mex interface of MATLAB. The solvers used are GLPK 4.46

[35], Gurobi 5.60 [40], and TOMLAB/CPLEX 12.5 [81]. Box plots[3] of the computation times for the 500 scenarios for the implicit and explicit MILP problem are given in Figure 4.5. Statistics on the computations are given in Table 4.3 for CPLEX, Gurobi, and GLPK. The mex interface of GLPK did not provide any solver statistics except for the computation time. The statistics that are given are the number of simplex iterations and the computation time. The integrality gap is also given, which is not a solver statistic, but a statistic of the MILP problem. It is the objective value of the optimal solution of the MILP problem divided by the objective value of the optimal solution of the linear programming relaxation of the MILP problem. The minimum value of the integrality gap is one, since the objective value of the optimal solution of the MILP problem can never be lower than the objective value of the optimal solution of the linear programming relaxation of the MILP problem. In general it is assumed that if the integrality gap is closer to one the problem is easier to solve.

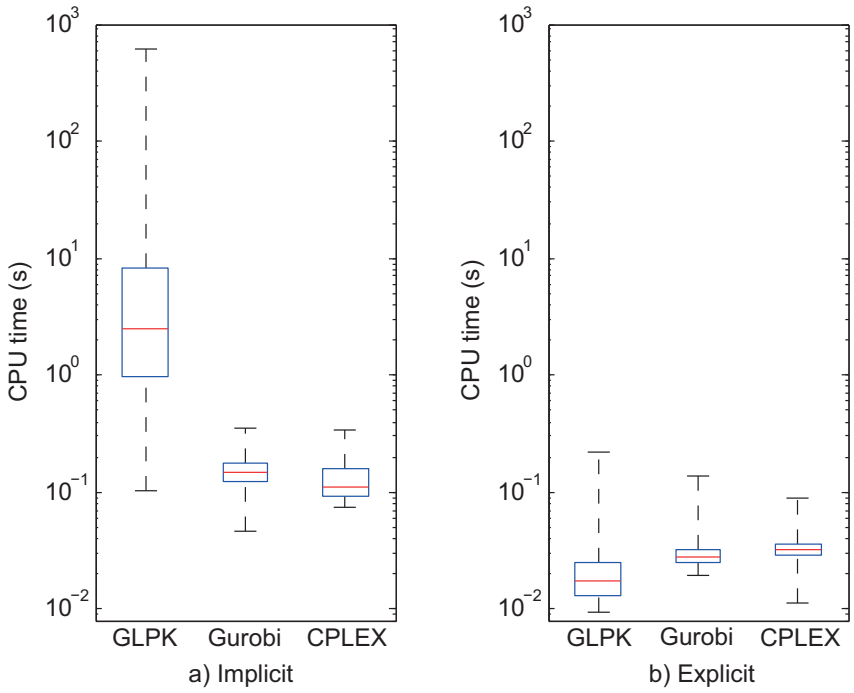

Figure 4.5: Computation time of the MILP solvers for the implicit and explicit MILP problem for the one hour prediction horizon.

For the GLPK solver the difference in computation time of the explicit MILP problem compared to the explicit MILP problem is the largest. The explicit MILP problem is solved 513 times faster on average than the MILP for the implicit model. For the Gurobi solver the difference is much smaller. The explicit MILP problem is solved only 4.69 times faster on average. On average the Gurobi solver needs to perform 3.26 times less

[3]Box plots divide the results into four equally sized parts: the 25% of the results with the lowest values are indicated by the lower vertical dashed line and bottom horizontal solid line. The 25% of the results with the highest values are represented by the upper vertical dashed line and top horizontal solid line. The other 50% is shown in the (blue) rectangle between the two dashed vertical lines, where the median of the results is presented by the (red) horizontal line splitting the box in two.

Table 4.3: Computation statistics for the given MILP solvers for the one hour prediction horizon.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Comp. time (s) (GLPK)	11.93	0.1014	603.92	0.0233	0.0096	0.2161
Comp. time (s) (Gurobi)	0.1477	0.0459	0.3494	0.0315	0.0196	0.1358
Comp. time (s) (CPLEX)	0.1266	0.0754	0.3400	0.0331	0.0112	0.0902
Simplex iter. (Gurobi)	159.72	17	869	49.06	0	411
Simplex iter. (CPLEX)	119.7	14	617	27.97	0	232
Integrality gap	1.0813	1.0148	1.2158	1.0715	1.0135	1.1757

simplex iterations. For the CPLEX solver we see a similar picture. The computation time is on average about 3.83 times faster. The number of simplex iterations that need to be performed is on average 4.28 times higher for the implicit MILP problem. The distance of the integrality gap to the value one is 12.1% lower for the explicit model compared to the implicit model (0.0715 compared to 0.0813), this is likely due to the reduction method, that simplifies the explicit problem. When we compare the fastest implicit solver (CPLEX) with the fastest explicit solver (GLPK), then the computation time needed to solve the implicit MILP problem is 5.44 times higher.

In some cases the solvers solving the optimization problem using the explicit model structure did not perform any simplex operations. This is likely due to the scenarios that had no reduction in avoidable delays. In that case the provided initial solution was the optimal solution and the solvers could prove optimality without performing any more simplex operations. The initial solution that is provided is the solution found when no control actions are taken.

Case study 1.2: A prediction horizon of two hours

In this case study we consider the same 500 scenarios as in the previous case study but now the model predictive controller has a prediction horizon of two hours to determine the dispatching actions that minimize the sum of all delays. The conversion into and reduction of the explicit model is done off-line and takes around 4 minutes to complete for the prediction horizon of two hour. For the implicit and explicit MILP problems based on the prediction horizon of two hours the average, minimum, and maximum number of continuous variables, binary variables, and constraints for are given in Table 4.4.

For the prediction horizon of two hours we will only look at the computation time and solver statistics, since the reduction of delays and the distribution of the delays are similar to those shown for the prediction horizon of one hour, the reduction is just a bit higher, and there are a few more smaller delays. The computation times for the 500 scenarios for the implicit and explicit MILP problem are given as box plots in Figure 4.6.

Statistics on the computation time for CPLEX, Gurobi, and GLPK are given in

Table 4.4: Number of constraints, continuous and binary variables of the MILP problems for the two hour prediction horizon.

	mean	min	max
Continuous	1543	1530	1559
Binary	2348	2307	2400
Constraints-Implicit	14037	13870	14233
Constraints-Explicit	50981	46044	57332

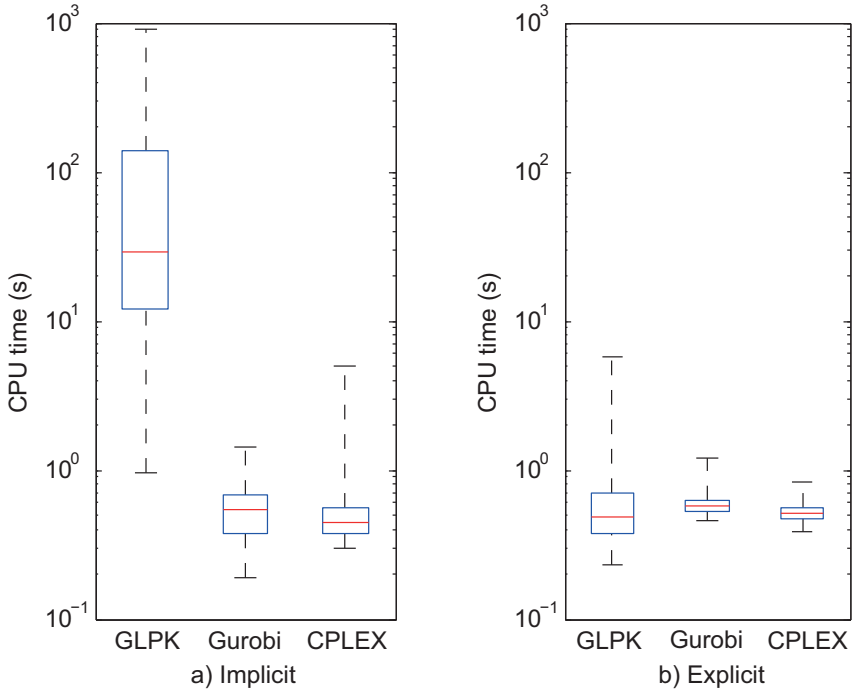

Figure 4.6: Computation time of the MILP solvers for the implicit and explicit MILP problem for the two hour prediction horizon.

Table 4.5.

In this case the average computation time of the explicit model for GLPK is 254 times lower, while the maximum is 156 times lower. For Gurobi the explicit model is solved 1.09 times slower on average. The number of simplex iterations is much higher for the explicit model. The increased computation time and number of simplex iterations is due to the increased size of the problem. The number of constraints is, on average, 3.63 times higher for the explicit model. For CPLEX the number of simplex iterations is lower for the explicit model, but the average computation time is still 1.04 times higher. This is again due to the increased size of the constraint matrix and as a result the simplex iterations take more time to complete. The maximum computation time however is 6.15 times lower for CPLEX when solving the explicit MILP problem. The distance of the integrality gap to the value one is 16.0% lower for the explicit model compared to the implicit model (0.100 compared to 0.119)

Table 4.5: Computation statistics for the given MILP solvers for the two hour prediction horizon.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Comp. time (s) (GLPK)	155.8	0.9599	903.7	0.6132	0.2286	5.792
Comp. time (s) (Gurobi)	0.5451	0.1870	1.4355	0.5943	0.4515	1.1903
Comp. time (s) (CPLEX)	0.5019	0.3007	5.0470	0.5247	0.3900	0.8200
Simplex iter. (Gurobi)	188.9	39	1175	525.2	0	1633
Simplex iter. (CPLEX)	242.2	55	923	71.98	0	262
Integrality gap	1.119	1.022	1.267	1.100	1.021	1.228

4.4.2 Case study 2: Minimization of the sum of arrival delays

When considering the delay in the network it can make more sense to only consider one delay per train at each station, so only the arrival or the departure delay at the station. Since passengers are mostly interested in the time they arrive we will consider minimizing the sum of arrival delays as the cost function.

In this case the cost function of the MILP problem becomes:

$$c^\top(\kappa)\, z(\kappa) = \begin{bmatrix} \mathbf{0}^\top & \mathbf{1}^\top & 0.0001 \times \mathbf{1}^\top \end{bmatrix}^\top \begin{bmatrix} d(\kappa) \\ a(\kappa) \\ \upsilon(\kappa) \end{bmatrix}.$$

where $d(\kappa)$ and $a((\kappa))$ are the departure and arrival time in $\chi(\kappa)$ respectively.

We do not need the departure delays and therefore we can simply calculate the arrival delay. For the explicit MILP problem this means that the constraint matrix A only needs to consist of the constraints from (4.8), and the constraints from the reduction method and infeasible train orders. This effectively reduces the size of the explicit MILP problem. To test the effects of the reduced number of constraints on the computation time of the solvers we have generated 250 new scenarios, using the same parameters as in the previous case study, and have compared the computation time needed to solve the implicit and explicit MILP problems again for one and two hour prediction horizons. In this case study we will only look at the computation time and computational statistics of the solvers, since the distribution and reduction of the delays are again similar to the distribution and reduction in the first case study.

Case Study 2.1: A prediction horizon of one hour

The specifications of the MILP problems for a prediction horizon of one hour are given in Table 4.6.

The explicit MILP problem has about half the number of continuous variables since it only needs to determine the arrival delays thanks to the explicit model structure; the implicit MILP problem has to determine all delays since the arrival delays depend on the arrival and departure delays of other trains through the implicit constraints. Because of

Table 4.6: Number of constraints, continuous and binary variables of the MILP problems.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Continuous variables	774	759	792	385	378	393
Binary variables	671	639	710	671	639	710
Constraints	5950	5794	6252	2210	1954	3061

the reduction method and the lower number of continuous variables the explicit MILP problem needs to consider, the number of constraints is on average 2.69 times lower than the number of constraints of the implicit MILP problem. We therefore expect the explicit MILP problem to be solved faster than the implicit MILP problem. The computation time needed to solve the implicit and explicit MILP problem for the three solvers is shown in Figure 4.7. The computation time, number of simplex iterations, and the integrality gap for the different MILP problems for CPLEX, Gurobi, and GLPK are given in Table 4.7.

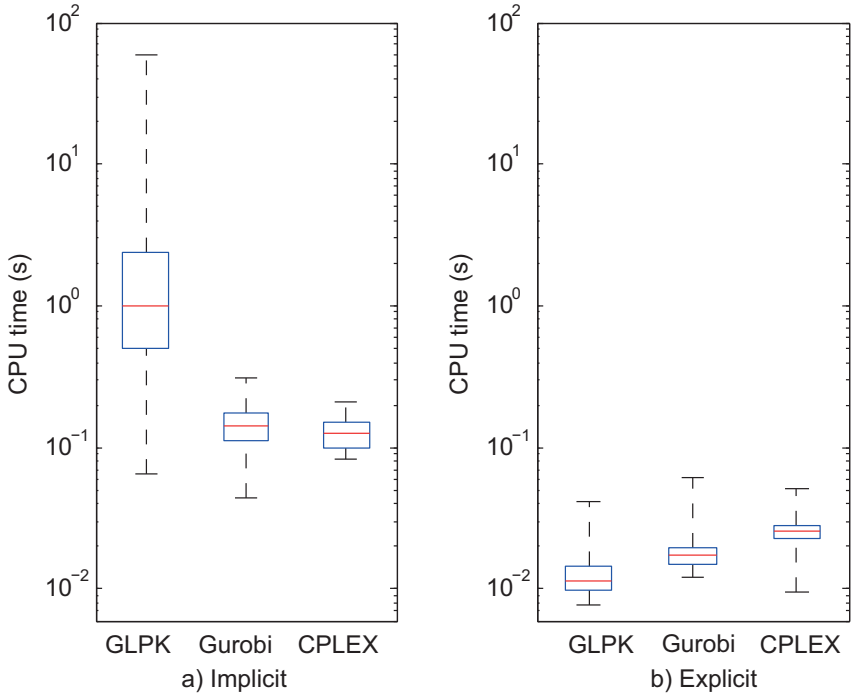

Figure 4.7: Computation time of the MILP solvers for the implicit and explicit MILP problem for the sum of arrival delays for the prediction horizon of one hour.

From these results we can conclude that the difference in computation time between the explicit and implicit MILP problem is the largest for GLPK. The explicit MILP problem is solved 272.7 times faster on average. For Gurobi the difference is much smaller,

Table 4.7: Computation statistics for the given MILP solvers for the prediction horizon of one hour.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Comp. time (s) (GLPK)	3.4993	0.0648	59.03	0.0128	0.0413	0.0078
Comp. time (s) (Gurobi)	0.1438	0.0438	0.3109	0.0122	0.0173	0.0610
Comp. time (s) (CPLEX)	0.1284	0.0836	0.2090	0.0259	0.0095	0.0506
Simplex iter. (Gurobi)	168.18	17	524	20.84	0	162
Simplex iter. (CPLEX)	103.39	14	331	15.27	0	66
Integrality Gap	1.098	1.026	1.181	1.086	1.023	1.169

but still significant, with the implicit MILP problem being solved 7.8 times slower on average. The number of simplex relaxations that need to be solved is on average 8.1 times higher for the implicit model. With CPLEX the implicit MILP problem is solved 5.0 times slower than the explicit MILP problem on average. The difference in the number of simplex relaxations that need to be solved is on average 6.8 times lower for the explicit model. When we compare the fastest implicit MILP solver (CPLEX) with the fastest explicit MILP solver (GLPK) the solution is found 10.0 times faster on average using the explicit MILP problem with GLPK. The distance of the integrality gap to the value one is 11.8% lower for the explicit model compared to the implicit model (0.086 compared to 0.098)

Case Study 2.2: A prediction horizon of two hours

For the two hour prediction horizon the number of constraints, continuous and binary variables of the MILP problems are given in Table 4.8.

Table 4.8: Number of constraints, continuous, and binary variables of the MILP problems for the prediction horizon of two hours.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Continuous variables	1543	1530	1559	769	762	776
Binary variables	2347	2307	2382	2347	2307	2382
Constraints	14035	13870	14212	28791	26208	32092

Due to the increased size of the prediction horizon the number of constraints, continuous, and binary variables have increased. The number of constraints of the explicit MILP problems is now 2.05 times higher than the number of constraints in the implicit model. This will affect the computation time of the solvers of the explicit MILP problems. The computation time needed to solve the implicit and explicit MILP problem for the three solvers is given in Figure 4.8. The computation time, number of iterations and

number of nodes explored for the different MILP problems for CPLEX and Gurobi are given in Table 4.9.

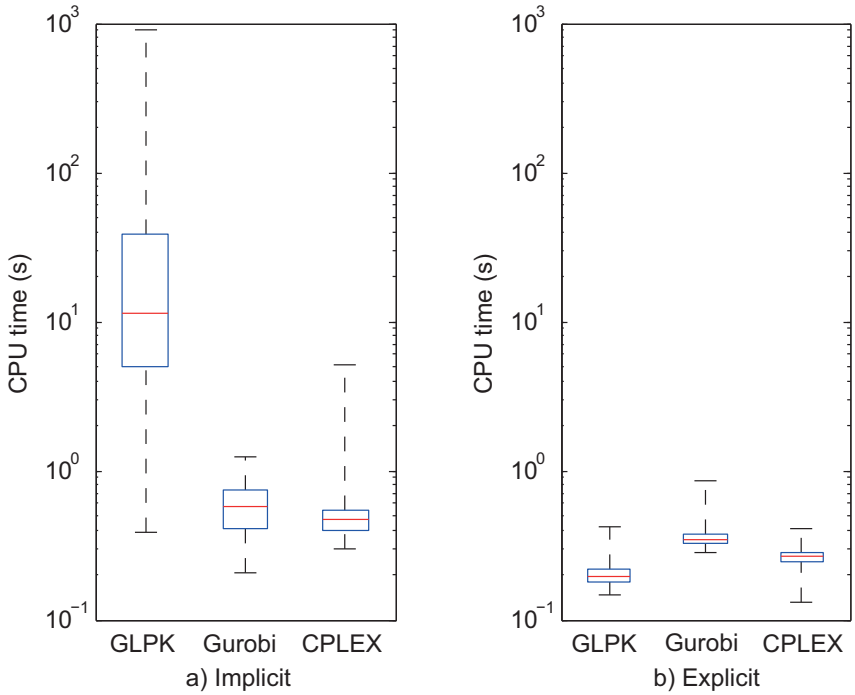

Figure 4.8: Computation time of the MILP solvers for the implicit and explicit MILP problem for the sum of arrival delays for the prediction horizon of two hours.

Table 4.9: Computation statistics for the given MILP solvers.

	Implicit			Explicit		
	mean	min	max	mean	min	max
Comp. time (s) (GLPK)	65.58	0.3904	902.7	0.2043	0.1484	0.4157
Comp. time (s) (Gurobi)	0.5906	0.2041	1.2246	0.3552	0.2862	0.8465
Comp. time (s) (CPLEX)	0.5213	0.3011	5.1109	0.2656	0.1328	0.4099
Simplex iter. (Gurobi)	227.0	31	1069	275	0	608
Simplex iter. (CPLEX)	224.5	46	642	33.42	0	128
Integrality gap	1.137	1.001	1.320	1.115	1.000	1.234

The GLPK solver solves the explicit MILP problems the fastest. It solves the explicit MILP problems 321 times faster on average than the implicit MILP problems. Gurobi solves the explicit MILP problems 1.66 times faster than the implicit MILP problems on average. CPLEX solves the explicit MILP problems 1.96 times faster than the implicit MILP problems on average. The fastest solver for the implicit MILP problems is CPLEX with an average computation time of 0.5213 seconds. The fastest solver for the explicit MILP problems is GLPK with an average computation time of 0.2043 seconds. The

explicit MILP is thus solved 2.55 times faster than the implicit MILP. This may be explained by the lower integrality gap of the explicit model. The distance of the integrality gap to the value one is 16.1% lower for the explicit model compared to the implicit model (0.115 compared to 0.137).

4.5 Case studies: MPC versus DMPC

In this section we compare the performance of the centralized MPC method to the four DMPC methods called DMPC method 1, 2, 3, and 4 that we introduced in Section 4.3. We use a model of the part of the Dutch railway network that is used by the Nederlandse Spoorwegen with the timetable of 2011[4]. The train lines, the line type, and their frequencies are shown in Appendix A. In the model 66 stations and/or junctions are considered where the order of the trains can be changed. There are 180 tracks connecting the stations and junctions. Per hour 326 trains traverse the network. A 12% buffer time on all running times is assumed. For the dwell times between 0 and 2 minutes buffer time is assumed, and at stations where a train must turn between 10 and 30 minutes buffer time is assumed. The headway times are based on norms and are between 3 and 5 minutes.

We use the same computer as in the previous case study with an AMD Phenom II X4 960T at 3.00GHz with 16GB memory running 64bit Windows 7 with MATLAB 2013b and we have solved the optimization problems with Gurobi 5.6.0.

Using the method described in Section 4.3 we have determined partitions of 2, 3, 4, 6, and 8 parts for the DMPC methods. The partitions can be seen in Figure 4.9 (b)-(f). The structure of the constraint matrix can be seen in Figure 4.10.

The case study will consists of two parts.

4.5.1 Case study 3: MPC versus DMPC part 1

For the first part we will compare the solution quality of the four DMPC approaches to each other and the centralized MPC approach for 1000 scenarios. The cost function of the centralized MPC is the sum of all delays: $c(\kappa)^\top z(\kappa) = [\mathbf{1}^{1\times n_x(\kappa)} \;\; 0.0001\cdot\mathbf{1}^{1\times n_\upsilon(\kappa)}] \begin{bmatrix}\chi(\kappa)^\top \\ \upsilon(\kappa)^\top\end{bmatrix}$, where $\mathbf{1}^{1\times m}$ is a 1 by m vector containing only ones, $n_x(\kappa)$ is the number of continuous variables at time instant $t(\kappa)$, and $n_\upsilon(\kappa)$ is the number of binary variables at time instant $t(\kappa)$. In this case study we only used the partition consisting of four parts and a prediction horizon of 60 minutes is used. For each scenario we generate delays in the first hour of the railway traffic and the controller will be activated after the first hour. No new delays are introduced after the first hour when the controllers are active. We delay 10% of the

[4]The complete timetable is too large to include in this paper and is no longer available online Since there have been very few major changes in the timetable in the last years the reader can get a general idea of the timetable from the 2015 timetable. The timetable of 2015 can be found (in Dutch) at http://www.ns.nl/reizigers/reisinformatie/informatie/informatie-tijdens-uw-reis/download-dienstregeling-2014-2015.html.

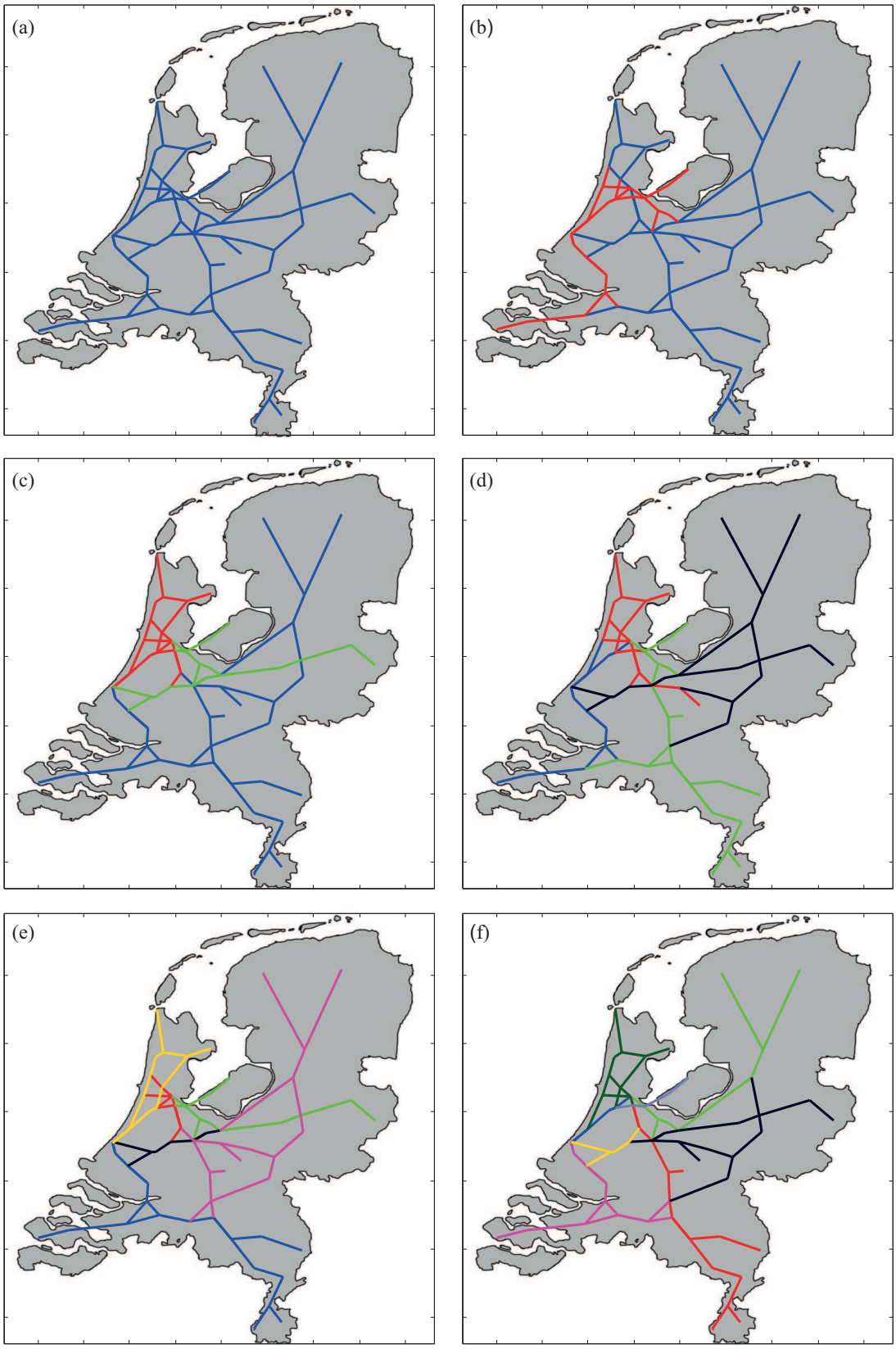

Figure 4.9: Model of the Dutch Network (a), and the partitions used for DMPC, partitioned in two (b), three (c), four (d), six (e), and eight (f) parts.

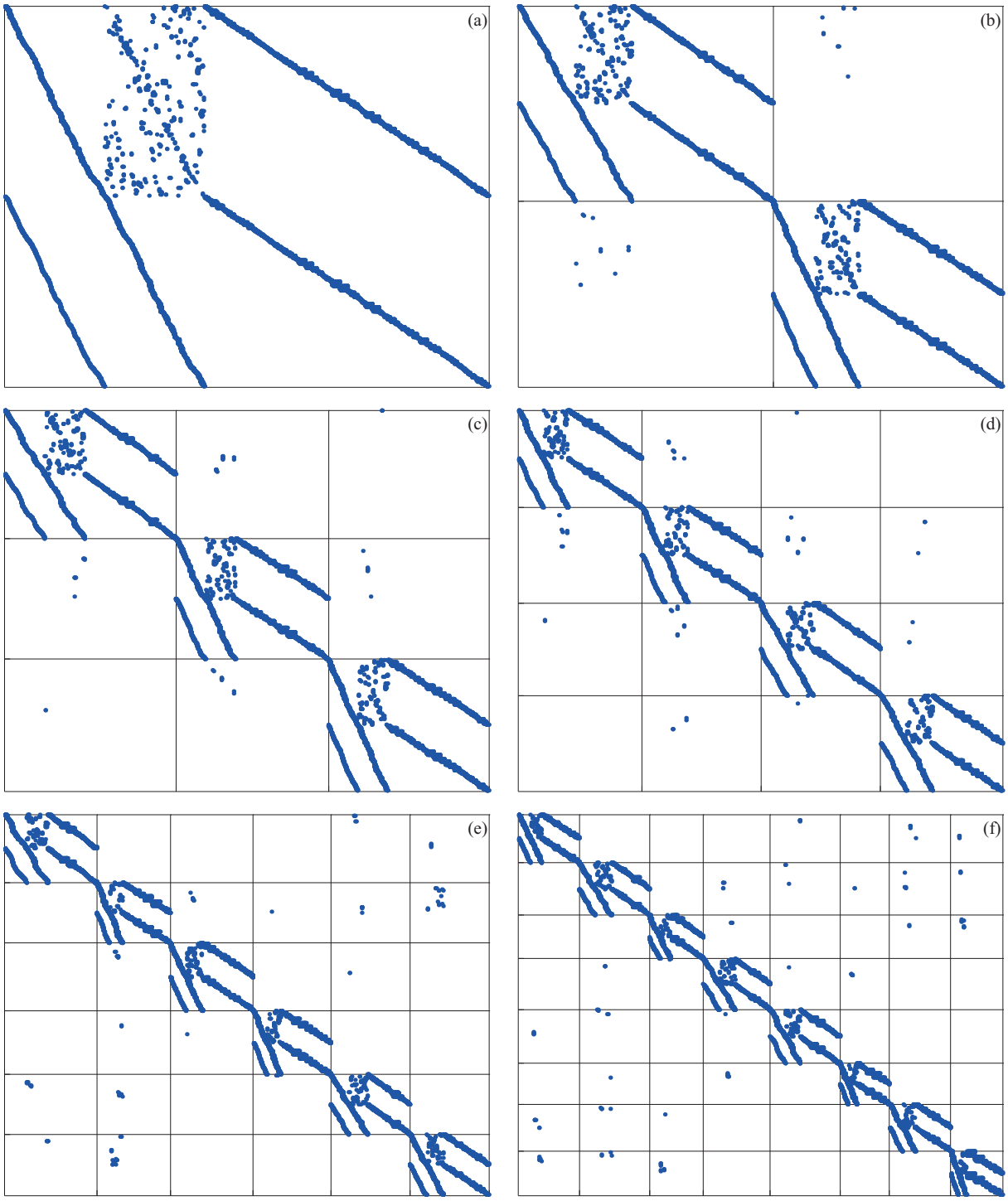

Figure 4.10: Structure of the constraint matrix for the entire network (a), and the partitions used for DMPC, partitioned in two (b), three (c), four (d), six (e), and eight (f) parts.

trains with a randomly generated delay according to a Weibull distribution with scale parameter 5 and shape parameter 0.8. We will only look at the first time instant after the first hour, so time instant $t(\kappa) = 60$, in which the controllers have to determine the optimal new schedule for the next hour based on the current situation of the railway network and traffic.

The relative increase in the sum of all delays because of the use of the DMPC approaches is shown in Figure 4.11. The absolute increase in the sum of all delays is shown in Figure 4.12. The computation time for the global model predictive controller and the four DMPC approaches to find their solution is shown in Figure 4.13.

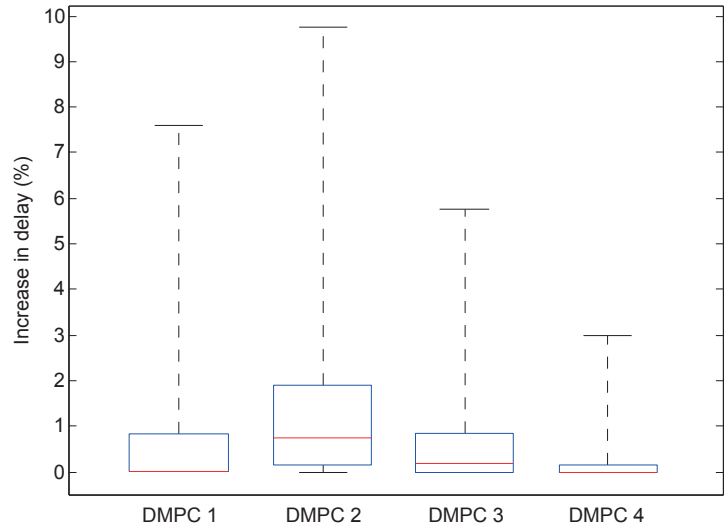

Figure 4.11: Relative increase in delays (%) compared to the solution of the global MPC, DMPC 1, DMPC 2, DMPC 3, and DMPC 4.

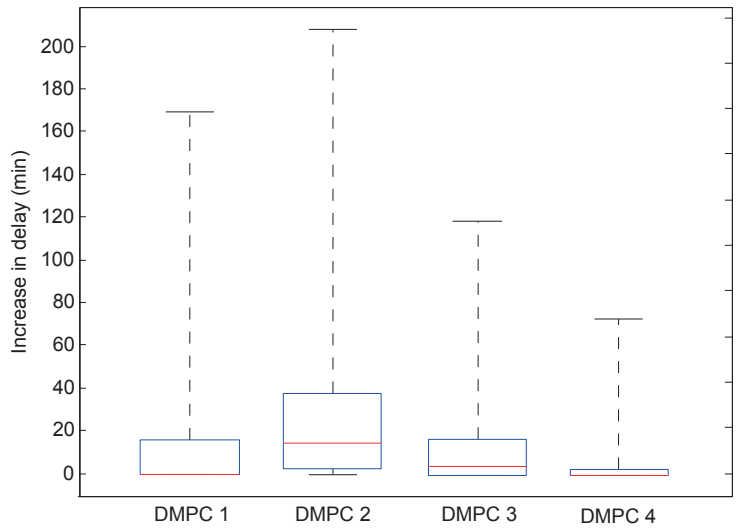

Figure 4.12: Absolute increase in delays (min) compared to the solution of the global MPC, DMPC 1, DMPC 2, DMPC 3, and DMPC 4.

From Figures 4.11 and 4.12 it is clear that the adjustment of the weights in DMPC methods 3 and 4 has a beneficial effect on the solution quality. Especially DMPC 4 finds

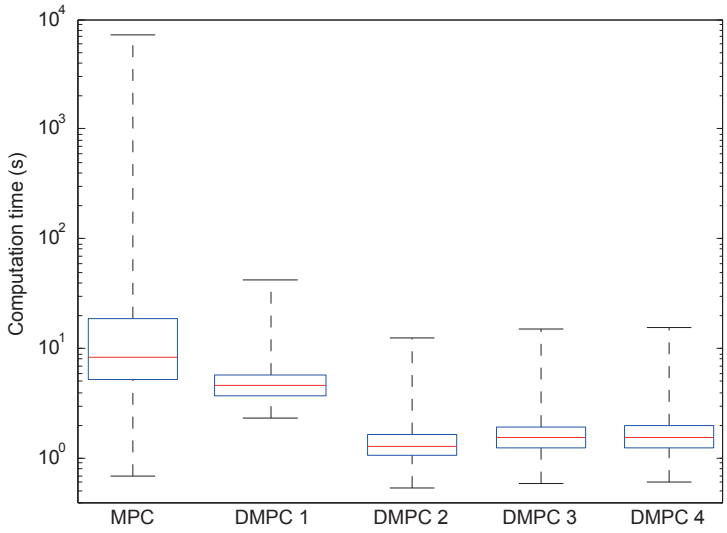

Figure 4.13: Computation time of the global MPC, DMPC 1, DMPC 2, DMPC 3, and DMPC 4.

very good solutions; the average increase in delays is only 0.14% (2.9 minutes), which is much lower than the average of 0.63% (12.7 minutes) for DMPC 1, and the 1.27% (25.1 minutes) average increase of DMPC 2. From Figure 4.13 it is clear that the that the adjustment of the weights does not have a significant effect on the computation time. DMPC 3 and 4 are slightly slower than DMPC 2, but still much faster than DMPC 1 and the global MPC approach.

4.5.2 Case study 4: MPC versus DMPC part 2

For the second part we have generated 100 scenarios, in each scenario the scheduled arrival and departure times for a 3 hour window are considered. The controllers have to update the schedule every minute and are in a closed loop such that the control actions each time instant are implemented and the consequences of those control actions affect the current and future time instants. The closed loop continues until all trains drive according to the normal schedule again. As a result between 180 and 200 optimizations are done per scenario depending on the delays. The same parameters are used for the delays of the trains, but this time the delays are added in between time instants while the trains are running. All partitions are considered and prediction horizons of lengths of 30, 45, 60, and 75 minutes are considered. For each scenario we have solved the rescheduling problem for the centralized MPC method and the four DMPC methods for the five different partitions for prediction horizons of length 30, 45, 60, and 75 minutes.

We assume the system initially has no delays, but that during operation delays occur. Therefore there are no delays at the start, and the first delays are added during the closed loop operation. As a result during the first 30 minutes the amount of delays increases, after that it stabilizes. Therefore, we do not consider the computation times of the optimizations for the first 30 minutes. Depending on the length of the prediction horizon the size of the optimization problems decreases near the end of the 3 hour window, and therefore we only consider the computation times up to the 130th minute, leaving us with

100 computation times for each scenario, or 10000 computation times in total, for each method, and each partition.

For methods based on MPC at each step the solution, or control input, should be determined before the new information is received and the next step starts. The implementation may take longer and depending on the time needed for the implementation the MPC method can be adjusted. In this case study we assume we get new information every minute and therefore the schedule is recomputed every minute. Because of that the computation time should be well below one minute, an acceptable maximum computation time would be 20 seconds, but lower would be better since the local controllers in our framework also have to compute the local schedules and routes based on the information our global controller provides.

Computation times

First consider the computation times of the centralized MPC method as given in Figure 4.14. Most of the time the new schedule is computed well within 20 seconds, but for the prediction horizons of 45, 60, and 75 minutes, some computation times are already above 60 seconds with a maximum of 158 seconds for the prediction horizon of 45 minutes. For a prediction horizon of 60 minutes this increases to 736 seconds and for a prediction horizon of 75 minutes the maximum becomes 1009 seconds. Because of the maximum computation time the implementations of the centralized MPC with a prediction horizon of 45, 60, and 75 minutes are currently not suitable for on-line railway traffic management.

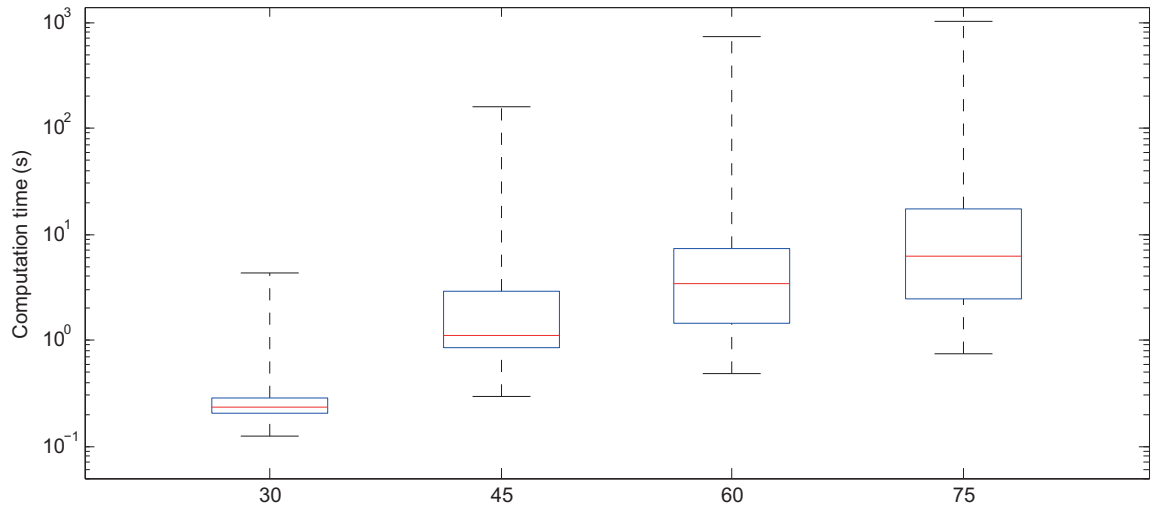

Figure 4.14: Computation time needed for the centralized model predictive control approach with a prediction horizon of 30, 45, 60, and 75 minutes.

Next consider the computation times of DMPC method 1 as shown in 4.15 and given in Table 4.10. For the prediction horizons of 30 minutes increasing the number of parts in the partition above three parts only increases the computation time. This is because

solving the optimization problem of the centralized MPC method is already relatively easy, therefore the computation time for each subproblem of the DMPC method is only slightly shorter than the centralized MPC method and with an increased number of parts in the partition the number of iterations the DMPC method needs to perform increases, resulting in an increase in computation time instead of a decrease. Since the optimization problem becomes harder to solve for larger prediction horizons, the number of parts for which the computation time still reduces is higher for larger prediction horizons. Only for prediction horizons of 60 and 75 minutes with a partitioning into two parts the maximum computation time goes above 60 seconds, with a maximum of 67 seconds for a prediction horizon of 60 minutes and 2 parts and 103 seconds for a prediction horizon of 75 minutes and 2 parts. For a prediction horizon of 75 minutes the maximum computation time is above 20 seconds for all partitions. For a prediction horizon of 60 minutes only the partition with 2 parts has a maximum computation time above 20 seconds. For the prediction horizons of 30 and 45 minutes the maximum computation time is below 20 seconds for all partitions.

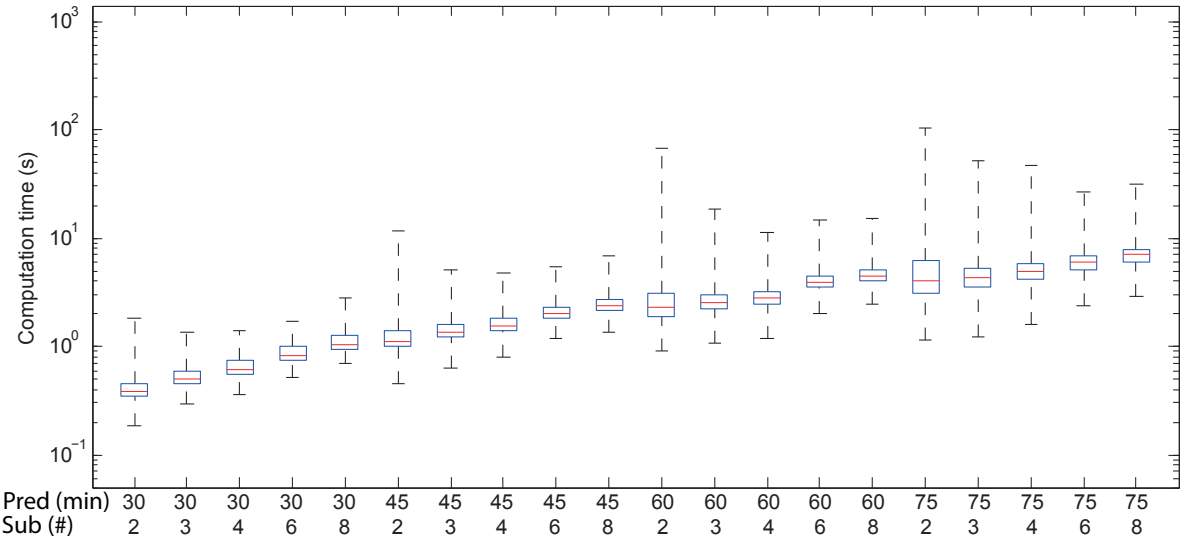

Figure 4.15: Computation time needed for DMPC method 1 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts.

Since DMPC methods 2, 3, and 4 are similar methods, their computation times are also similar, as shown in Figures 4.16-4.18 respectively. We will therefore discuss the computation times of these three methods at the same time. Compared to the computation times of DMPC method 1 there are clear differences. For DMPC methods 2, 3, and 4 increasing the number of parts in the partition decreases the computation time for all lengths of the prediction horizon tested and all partitions, except for DMPC 4, with a prediction horizon of 60 minutes and eight parts in the partition and with a prediction horizon of 75 minutes and four parts in the partition. In both of these cases the maximum computation time is slightly higher than for some of the other partitions with less parts. The cause of the decrease in computation time is that each subproblem only contains of a part of the constraints of the centralized problem with no overlap in constraints of the

Table 4.10: Computation time for the MPC and DMPC methods for Case Study 3.

	30 min			45 min			60 min			75 min		
	avg	min	max	avg	min	max	avg	min	max	avg	min	max
MPC	0.31	0.12	4.34	1.93	0.30	158	5.60	0.48	736	11.8	0.75	1009
DMPC1 (2)	0.41	0.19	1.79	1.24	0.45	11.8	2.88	0.90	67.4	5.50	1.16	103
DMPC1 (3)	0.53	0.30	1.34	1.46	0.62	5.05	2.78	1.06	18.5	4.73	1.22	52.4
DMPC1 (4)	0.66	0.35	1.39	1.65	0.81	4.71	3.01	1.18	11.3	5.22	1.57	47.5
DMPC1 (6)	0.89	0.52	1.72	2.15	1.18	5.44	4.17	1.99	14.5	6.20	2.34	26.8
DMPC1 (8)	1.12	0.69	2.85	2.54	1.36	6.92	4.80	2.42	15.1	7.32	2.86	32.0
DMPC2 (2)	0.23	0.09	1.06	0.86	0.23	12.4	1.99	0.38	64.9	3.47	0.54	118
DMPC2 (3)	0.20	0.08	0.56	0.70	0.22	3.20	1.43	0.37	12.2	2.24	0.44	28.1
DMPC2 (4)	0.19	0.08	0.46	0.53	0.18	2.40	1.03	0.31	5.57	1.70	0.35	13.0
DMPC2 (6)	0.17	0.08	0.40	0.45	0.17	1.63	0.78	0.25	4.95	1.20	0.34	7.54
DMPC2 (8)	0.16	0.08	0.38	0.38	0.15	1.05	0.64	0.23	2.38	0.98	0.25	5.84
DMPC3 (2)	0.23	0.08	1.02	0.85	0.23	16.5	1.99	0.46	120	3.42	0.54	43.9
DMPC3 (3)	0.20	0.08	0.93	0.70	0.22	3.24	1.47	0.37	13.0	2.32	0.44	24.5
DMPC3 (4)	0.19	0.08	0.58	0.53	0.18	2.03	1.06	0.31	5.59	1.75	0.40	11.3
DMPC3 (6)	0.17	0.08	0.56	0.45	0.17	1.79	0.79	0.25	3.41	1.22	0.31	9.24
DMPC3 (8)	0.16	0.08	0.36	0.37	0.15	1.44	0.64	0.23	3.36	0.97	0.24	7.37
DMPC4 (2)	0.23	0.08	1.17	0.86	0.23	12.6	2.06	0.46	75.6	3.49	0.54	69.6
DMPC4 (3)	0.20	0.08	0.90	0.67	0.22	4.89	1.47	0.37	10.6	2.39	0.45	19.0
DMPC4 (4)	0.19	0.08	0.49	0.53	0.18	2.74	1.09	0.28	6.07	1.82	0.32	21.7
DMPC4 (6)	0.17	0.08	0.57	0.44	0.16	2.58	0.79	0.25	5.48	1.25	0.33	10.5
DMPC4 (8)	0.16	0.08	0.34	0.37	0.15	1.06	0.63	0.23	9.82	0.94	0.24	5.81

subproblems. Thus increasing the number of subproblems reduces the number of constraints of each subproblem. This is in contrast with the subproblems of DMPC method 1, where each subproblem contains all constraints of the centralized problem. Therefore, increasing the number of subproblems does not reduce the number of the constraints of each subproblem.

For DMPC method 2 the maximum computation time is above 60 seconds in only two cases: a prediction horizon of 60 minutes and the partition into two parts with a maximum of 65 seconds, and a prediction horizon of 75 minutes and the partition into two parts with a maximum of 118 seconds. There is only one case where the maximum computation time is between 20 and 60 seconds: a prediction horizon of 75 minutes and the partition in three parts with a maximum of 24 seconds. All other combinations of partitions and prediction horizon lengths have a maximum computation time below 20 seconds and can be used for on-line railway traffic management.

For DMPC method 3 the maximum computation time is above 60 seconds in only one case: A prediction horizon of 60 minutes and the partition in two parts with a maximum of 120 seconds. There are two cases where the computation time is between

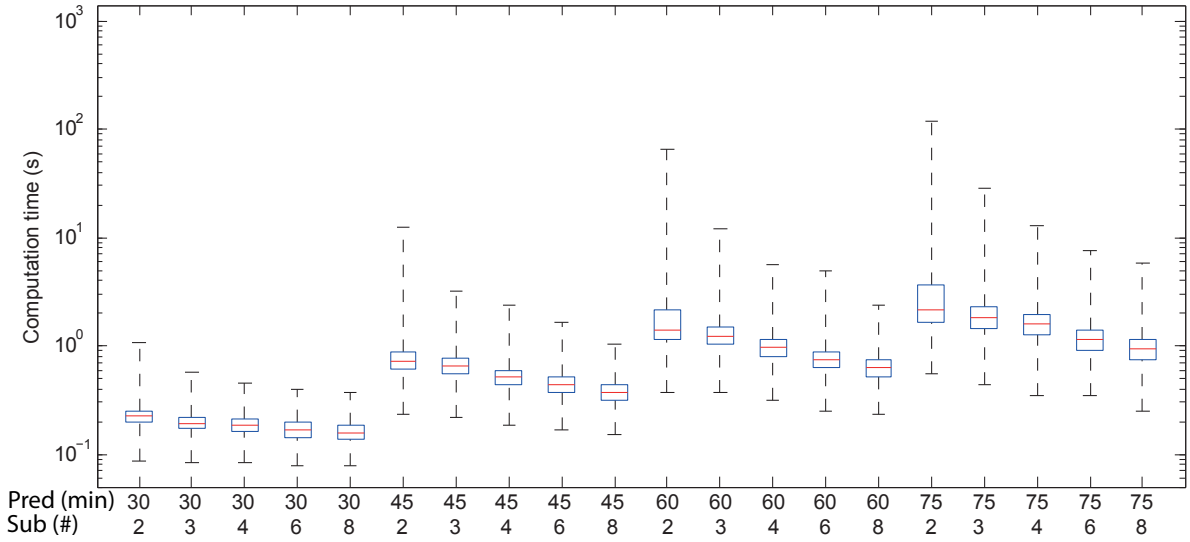

Figure 4.16: Computation time needed for DMPC method 2 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts.

20 and 60 seconds: A prediction horizon of 75 minutes and the partitions in two and three parts with maximum computation times of 44 and 24 seconds respectively. All other combinations of prediction horizon lengths and partitions are below 20 seconds and therefore suitable for on-line railway traffic management. It is unexpected that the maximum computation time of the prediction horizon of 75 minutes with the partition in two parts is lower than the maximum computation time with the prediction horizon of 60 minutes and the partition in two parts. The longer prediction horizon in general makes the problems harder to solve, but due to the binary nature of the problems there may be a few scenarios that are especially hard to solve and may take much longer than most. Therefore no strict conclusions can be drawn about the maximum computation time, which is some cases may still exceed the computation times we have found. We can only say that it is highly unlikely that the maximum computation time is higher than the maximum we have determined under the similar delay scenarios.

Finally for DMPC method 4 there are again two cases where the maximum computation time is above 60 seconds: the prediction horizon of 60 seconds with the partition in two parts with a maximum computation time of 76 seconds and the prediction horizon of 75 minutes with the partition in two parts with a maximum computation time of 70 seconds. Only for the case of the prediction horizon of 75 minutes and the partition in four parts is the maximum computation time between 20 and 60 seconds. For all other cases the maximum computation time is below 20 seconds.

Next we will discuss the delay reduction achieved with the various (D)MPC methods proposed.

Delay reduction

In order to determine the delay reduction achieved with the various control methods we compare the total delay of the 100 scenarios combined for all methods, prediction

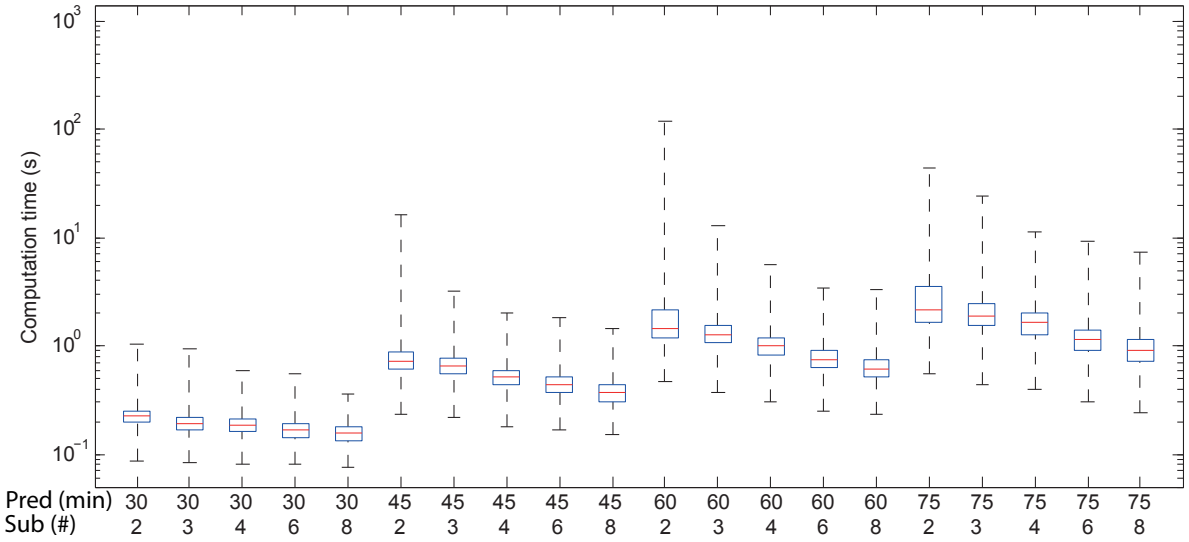

Figure 4.17: Computation time needed for DMPC method 3 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts.

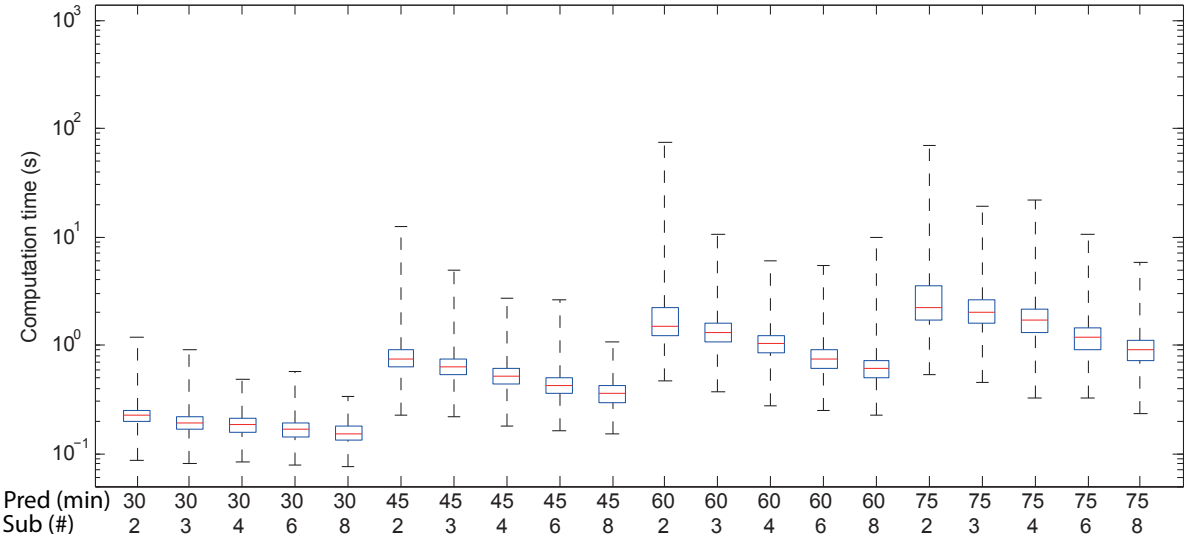

Figure 4.18: Computation time needed for DMPC method 4 with a prediction horizon of 30, 45, 60, and 75 minutes for the partitions in 2, 3, 4, 6, and 8 parts.

horizon lengths, and partitions to the nominal case. In the nominal case no train orders are changed: only the arrival and departure times are changed to avoid conflicts. We determine the reduction in delays compared to the nominal case for all cases in percent of the total delay of the nominal case. For the centralized MPC approach the delay reduction is given in the bar plot in Figure 4.19.

For the DMPC approaches the delay reduction is given in the bar plots in Figure 4.20 for the various prediction horizon lengths and partitions..

All the data is also given in Table 4.11.

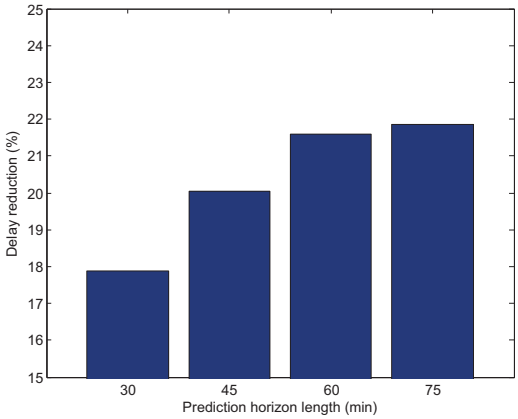

Figure 4.19: Reduction of the delays in percentage compared to the nominal case for the centralized MPC approach for Case Study 3.

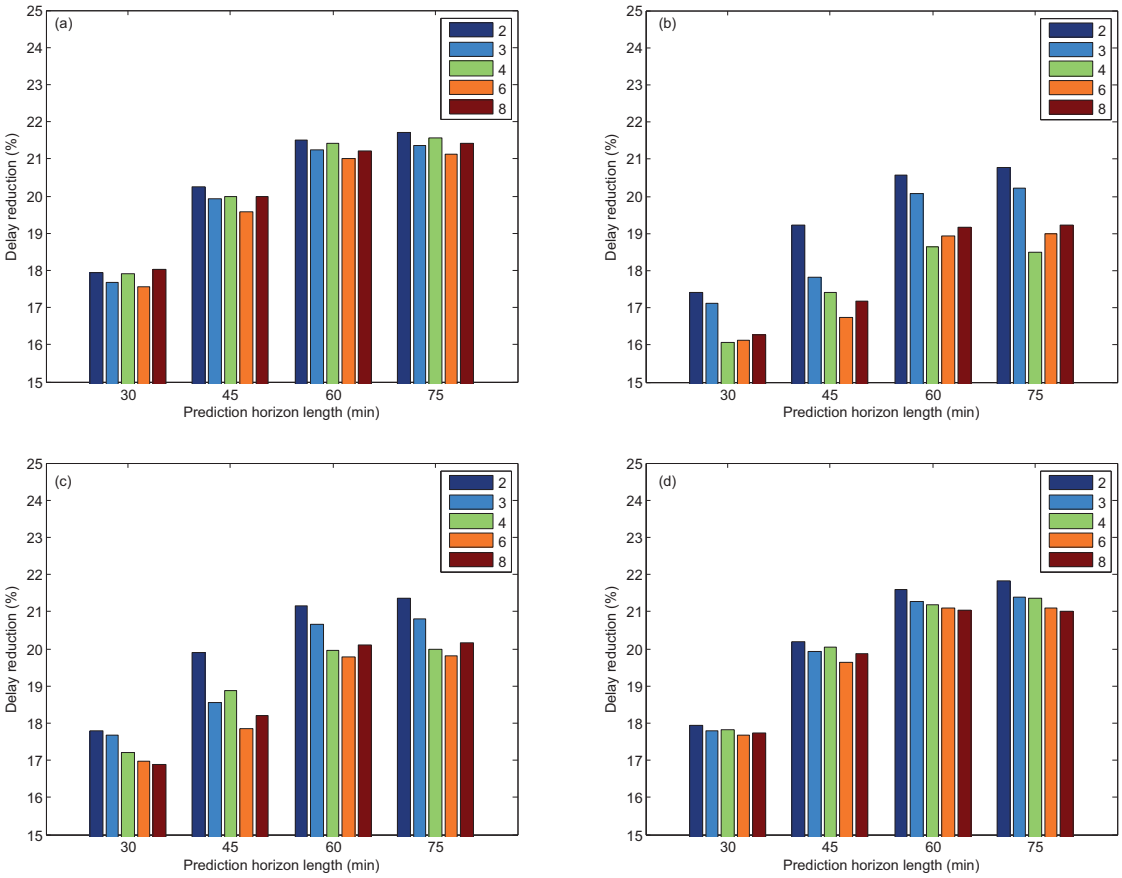

Figure 4.20: Reduction of the delays in percentage compared to the nominal case for (a) DMPC approach 1, (b) DMPC approach 2, (c) DMPC approach 3, and (d) DMPC approach 4 for the partitions in 2, 3, 4, 6, and 8 parts.

It is clear that DMPC method 4 always outperforms methods 2 and 3 in terms of

Table 4.11: Delay reduction in % for the centralized MPC and the DMPC methods.

	30 min	45 min	60 min	75 min
MPC	17.9%	20.0%	21.6%	21.9%
DMPC 1 (2)	17.9%	20.2%	21.5%	21.7%
DMPC 1 (3)	17.7%	19.9%	21.2%	21.4%
DMPC 1 (4)	17.9%	20.0%	21.4%	21.6%
DMPC 1 (6)	17.6%	19.6%	21.0%	21.1%
DMPC 1 (8)	18.0%	20.0%	21.2%	21.4%
DMPC 2 (2)	17.4%	19.2%	20.6%	20.8%
DMPC 2 (3)	17.1%	17.8%	20.1%	20.2%
DMPC 2 (4)	16.1%	17.4%	18.6%	18.5%
DMPC 2 (6)	16.1%	16.7%	18.9%	19.0%
DMPC 2 (8)	16.3%	17.2%	19.2%	19.2%
DMPC 3 (2)	17.8%	19.9%	21.2%	21.4%
DMPC 3 (3)	17.7%	18.5%	20.7%	20.8%
DMPC 3 (4)	17.2%	18.9%	20.0%	20.0%
DMPC 3 (6)	17.0%	17.8%	19.8%	19.8%
DMPC 3 (8)	16.9%	18.2%	20.1%	20.2%
DMPC 4 (2)	17.9%	20.2%	21.6%	21.8%
DMPC 4 (3)	17.8%	19.9%	21.3%	21.4%
DMPC 4 (4)	17.8%	20.1%	21.2%	21.4%
DMPC 4 (6)	17.7%	19.6%	21.1%	21.1%
DMPC 4 (8)	17.7%	19.9%	21.0%	21.0%

delay reduction. The difference in computation time between DMPC methods 2, 3, and 4 is negligible. Therefore, we will not consider DMPC methods 2 and 3 in the rest of the discussion.

In Figure 4.21 the delay reduction is plotted against the maximum computation for DMPC methods 1 and 4 and the MPC method. As is clear from this figure, the DMPC methods can achieve a similar delay reduction in a lot less time. In most cases DMPC method 4 is even slightly faster than DMPC 1 when comparing for similar reductions in delay.

Furthermore it is clear that by increasing the prediction horizon the delays are reduced more. The effects of increasing the prediction horizon further diminishes when the prediction horizon is longer: from 30 to 45 minutes the delay reduction for the centralized MPC is improved with 12.0%, from 45 to 60 minutes it is already slightly lower with an improvement of 7.8%, and from 60 to 75 minutes the improvement is only 1.2%. For the DMPC approaches we see similar results. An explanation for this is that the further ahead the controller predicts the future arrival and departure times, the less accurate those departure and arrival times become. As a result most of the future arrival and

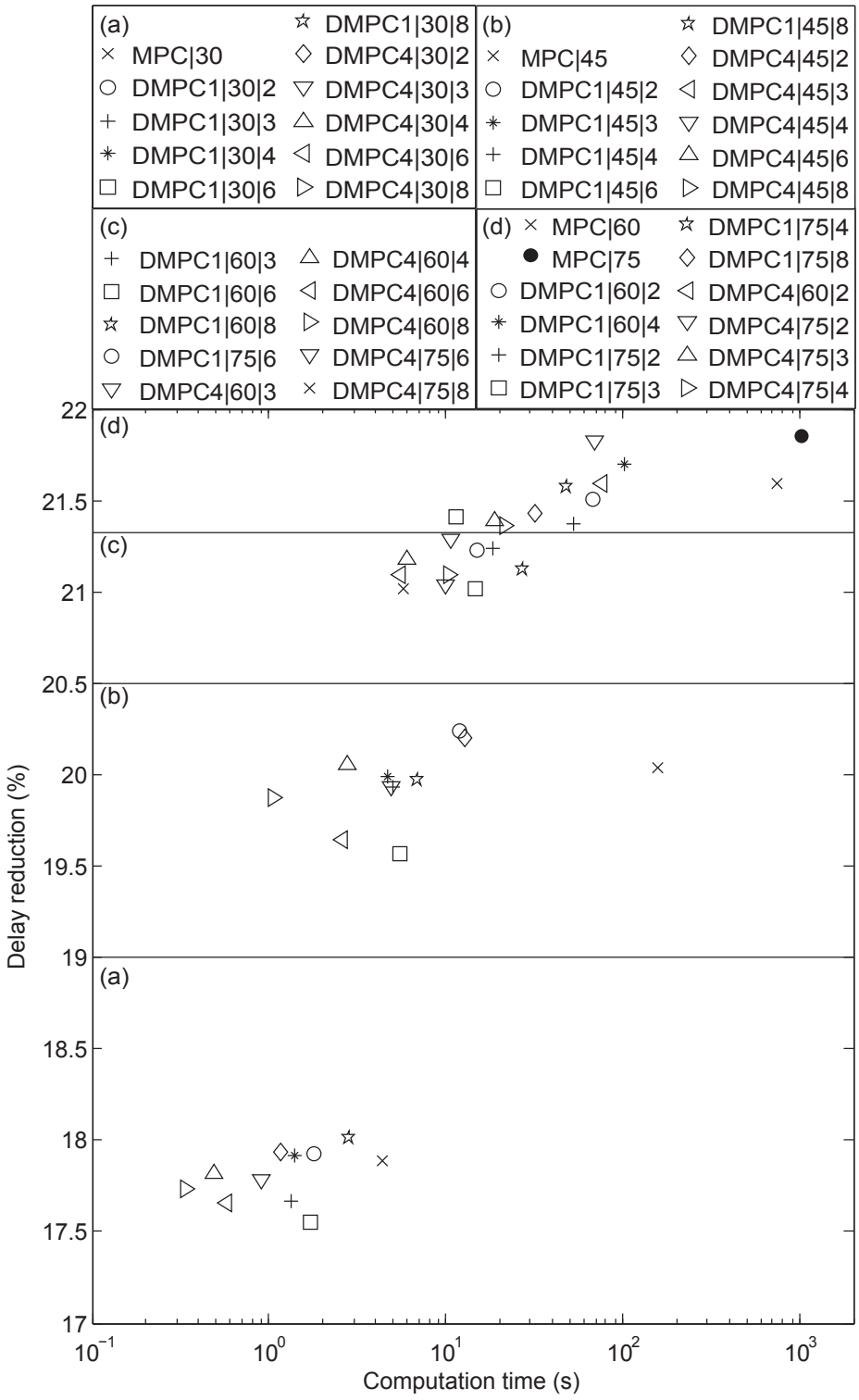

Figure 4.21: Reduction of the delays in percentage plotted against the maximum computation for DMPC method 1, DMPC method 4, and the MPC method for the partitions in 2, 3, 4, 6, and 8 parts, and the prediction horizons of 30, 45, 60, and 75 minutes.

departure times will be changed at future time instants when new information becomes available and better predictions can be made.

When we only consider the delay reduction and not the computation time, the centralized MPC with a prediction horizon of 75 minutes reduces the delays the most: 21.9%. Of the DMPC methods, DMPC method 4 with a prediction horizon of 75 minutes and partitioned into two parts has the highest delay reduction: 21.8%. The difference in delay reduction between the two is almost negligible, but the difference in computation time is substantial; the average computation time needed for DMPC method 4 is 3.38 times lower than for the centralized MPC method and the maximum computation time is 14.5 times lower.

When we consider a limit on the maximum computation time of 20 seconds DMPC method 1 with a prediction horizon of 60 minutes and partitioned into four parts and DMPC method 4 with a prediction horizon of 75 minutes and partitioned into three parts have the best results. DMPC method 4 has a slightly lower average computation time, but a higher maximum computation time. Both methods achieve a 21.4% reduction in delays, which is the highest of all methods with a maximum computation time below 20 seconds.

4.6 Summary

In this chapter the basics of MPC were explained and it has been shown how MPC can be applied to the problem of on-line railway traffic management using the switching max-plus-linear model explained in the previous chapter. At each time instant an optimization problem then needs to be solved to determine the dispatching actions that minimize the delays. If the cost function of the optimization problem is a linear function of the event times and control variables and the switching max-plus-linear model is rewritten into a set of mixed integer linear constraints a mixed integer linear programming problem is obtained.

A case study using a model based on the Dutch railway network, excluding regional trains, was conducted. The goal was to test the performance in terms of computation speed and delay reduction of the proposed MPC approach using the implicit and reduced explicit switching max-plus-linear model for prediction horizons of one and two hours for the sum of all delays and the sum of arrival delays. In all cases the MILP problem based on the reduced explicit model was faster, while achieving the same delay reduction, especially for the sum of arrival delays and a prediction horizon of 1 hour the explicit MILP was solved much faster; the average computation time was 10 times lower. For the other instances the difference was smaller. The computation time needed to convert and reduce the implicit model to the reduced explicit model took around 40 seconds for the model using a one hour prediction horizon. When the prediction horizon was increased to two hours the computation time increase to around 4 minutes. It may happen that for large models that include all trains the conversion takes much longer or becomes impractical to calculate. Further research is needed to determine this. Furthermore for larger delays

the reduction method is less effective and the benefits may be reduced further. But if the conversion from the implicit to the explicit model, and the reduction method, could be sped up such that it can be applied on-line the reduction method can be changed such that the limitation of the control actions is based on a maximum deviation from the last known solution. By making this change the solutions found with the model predictive controller using the reduced explicit model are optimal for larger deviations from the nominal timetable without increasing the problem complexity, as long as the change in situation in the network between time instants is limited.

For large instances the model predictive control problem can be too hard to solve. To overcome this problem four distributed model predictive control approaches were proposed. With distributed model predictive control the centralized problem is partitioned into smaller, faster to solve subproblems. These subproblems are then solved iteratively and the solutions are found are used to update the other subproblem until they converge to a global solution.

To test the proposed DMPC approaches two new case studies were performed. In the first case study a model of the Dutch railway network containing all Nederlandse Spoorwegen trains as described in the timetable of 2011 was used. The network was partitioned into four parts for the DMPC approaches. A prediction horizon of 60 minutes was used and the approaches were compared in terms of computation time and delay reduction for a single time-instant for 1000 scenarios. The result was that DMPC method 4 was the best by finding solutions that on average only had 1.27% more delays than the centralized MPC approach, while the computation time was much lower than for the centralized approach and among the lowest for the DMPC approaches.

In the second case study the centralized problem was partitioned into 2, 3, 4, 6, and 8 parts for the DMPC approaches and tested for prediction horizons of 30, 45, 60, and 75 minutes. Instead of comparing the performance of a single time instant for multiple scenarios, now the performance was compared for a three hour window, where the controller was in a closed loop, such that the control actions each time instant are implemented and the consequences of those control actions affect the current and future time instants. Therefore, each minute a new optimization had to be solved and implemented. The total delay in this three hour was compared for all four approaches, for all five partitions, and four prediction horizon lengths, for 100 scenarios. In the end the DMPC methods 1 and 4 performed comparable in both computation time and delay reduction and when considering limits on the maximum computation time outperformed the centralized MPC approach. The benefit of increasing the prediction horizon on the delay reduction lessens as the prediction horizon becomes larger, while the computation time grows exponentially. Based on the case studies we performed it seems that a prediction horizon with length between 60 and 75 minutes gives the best delay reduction compared to the computation time. There are several possible explanations for the lower increase in the delay reduction for increasing prediction horizons. First of all the predictions further ahead in the future are less accurate and more likely to change when new information becomes available in the future. A second explanation could be that due to the buffer times in dwell and running times, the delays are absorbed and for longer prediction horizons more delays will have

been absorbed at the end of the horizon and for smaller delays the controller is less likely to take control actions. Further research should be done on the effect of buffer times and the amount of delays on the delay reduction for large prediction horizons.

Chapter 5

Legged locomotion

This chapter gives an overview on current approaches for the modeling of locomotion patterns for legged locomotion namely central pattern generators and Buehler clocks. A max-plus approach is proposed that generalizes the Buehler clock approach and with the use of max-plus theories the transient and steady-state behavior is analyzed. Using the max-plus properties that have been derived optimal gait switches are derived and a method is proposed to let the robot accelerate or decelerate at a constant pace. This chapter is based on the papers of [67, 68]. The contributions in those papers that have been written by the author of this thesis are the main focus of this chapter. The contributions consist of the proof of the uniqueness of the eigenvector found in Section 5.3, optimal gait switching in Section 5.4, and the simulations of the max-plus-gait scheduler in Section 5.5.

5.1 Introduction

The controller for the legged locomotion of a robot can be structured in two subsystems: A gait reference generator and a tracking controller, as shown in Figure 1.3 from Chapter 1 and repeated here for convenience in Figure 5.1. The gait reference generator generates the cyclic reference signals that describe synchronization between the legs resulting in a cyclic motion. The dynamic tracking controller translates these reference signals into the actual movement the feet of the robot should make. In this chapter a novel approach for the gait reference generator is presented. The gait reference generator is described by a class of max-plus-linear systems that realize schedules for the touchdown and lift-off of the legs of a robot for a given class of gaits. Modeling the gait reference generator as a max-plus-linear system was first described in Lopes et al. [65, 66]. In this chapter closed-form expressions for the max-plus eigenvalue and eigenvector of the system matrix will be derived, and it will be shown that the max-plus eigen-parameters are max-plus unique, implying a unique steady-state behavior. The importance of having closed-form expressions and uniqueness of the max-plus eigenstructure is that, not only can one compute the following parameters very fast without having to run simulations or numerical algorithms (e.g. Karp's algorithm [3]), but one also has guarantees of uniqueness: the motion of the robot

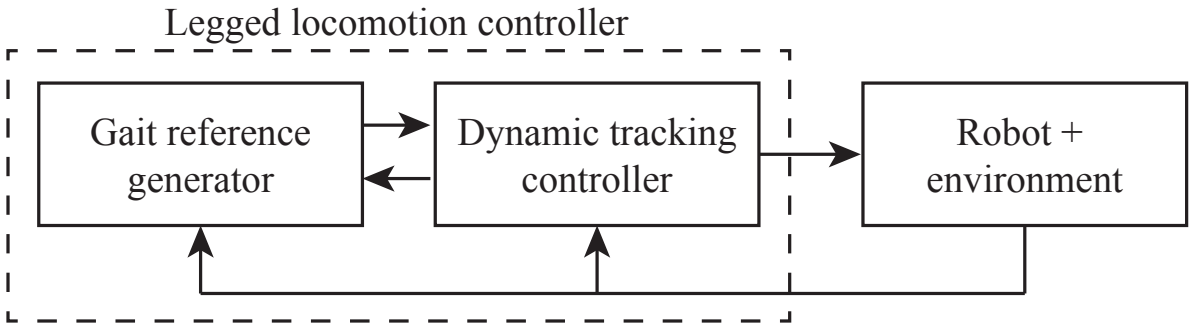

Figure 5.1: The standard partitioning of a legged locomotion controller. The gait reference generator subsystem provides reference signals to the tracking controller. Feedback can exist from both the robot and the tracking controller to the gait reference generator.

will always converge in a finite number of steps to the behavior described by the current gait and its parameters, even after gait switches or temporary disturbances[1]. The class of max-plus-linear systems also ensure *kinematic stance stability*[2] during disturbances and gait switches. Ensuring kinematic stance stability is fundamental when designing gait controllers for robotics. Additionally we present a bound on the number of cycles needed to reach steady-state motion after changing gaits.

Finally the results of simulations of the proposed legged locomotion controller are presented.

The chapter is built up as follows: in Section 5.2 a short literature survey is given on legged locomotion controllers and the approach proposed in this chapter is compared to the approaches in literature. In Section 5.3 several properties of the max-plus-linear system defining the gaits are derived. In Section 5.4 methods for determining optimal gait switches are presented. In Section 5.5 simulation using the max-plus gait reference generator are given. Finally in Section 5.6 the chapter is summarized, conclusions on the proposed approach are drawn, and recommendations for future research are given.

5.2 Modeling of legged locomotion

Central pattern generators (CPGs) are currently the standard tool for designing gait reference generators (see Ijspeert [48] for a survey on CPGs). CPGs are neural networks found in animals that can generate complex periodic signal patterns. They are called *central* pattern generators because they do not require sensory feedback to produce the patterns. In animals they generate rhythmic patterns for movement. So CPGs offer a natural bio-inspired control framework that addresses locomotion patterns.

[1]The disturbances considered in this chapter are delayed lift-off and touchdown times, due to temporary obstructions.

[2]In this thesis a robot is "kinematic stance-stable" if it can be guaranteed that there are always sufficient legs in stance to ensure the robot does not fall over.

Although widely used, CPGs offer their own set of challenges because of their mathematical formulation as sets of coupled differential equations. One of those challenges is the transient behavior that exists during gait transitions. Gait transitions are a very natural occurrence in nature; animals change gait to accommodate for different types of terrain, locomoting speeds, and to minimize the energy needed to move at the desired speed. As in normal systems modeled by differential equations, the transient behavior is typically less understood than the steady-state behavior. A lot of researchers have worked on gait transition in the CPG framework (see Aoi et al. [1], Daun-Gruhn and Toth [24], Inagaki et al. [49, 50], Li et al. [62], Nagashino et al. [72], Santos and Matos [79], Zhang et al. [95], and the references within [48]). Other work on gait transition without using CPGs in the continuous-time domain has been performed by Haynes and Rizzi [41], Haynes et al. [43]. The traditional approach for gait transition in the CPG framework exploits the bifurcations that occur when changing parameters in the set of coupled differential equations. This can lead to intricate analysis of the global behavior due to the continuous-time models used.

5.2.1 Central pattern generators

In robotics, CPGs are usually implemented by solving sets of coupled differential equations on-line [48]. An abstract phase $\theta_i \in \mathbb{S}^1$ is associated to each leg i representing its periodic motion, with $\mathbb{S}^1$ representing the circle. The dynamical equations for the full phase state $\theta = [\theta_1 \ \cdots \ \theta_n]^\top \in \mathbb{T}^n$ can be written as:

$$\dot{\theta}(\tau) = V + h(\theta(\tau)), \tag{5.1}$$

where $\mathbb{T}^n$ is the n-torus (the Cartesian product of n circles), $V \in \mathbb{R}^n$ represents the desired phase velocity vector, τ represents time, and the function h includes the desired coupling between each phase. A common realization of (5.1) is presented below [47]:

$$\dot{\theta}_i(\tau) = v + \sum_j w_{i,j} \sin(\theta_j(\tau) - \theta_i(\tau) - \phi_{i,j}), \tag{5.2}$$

where $v \in \mathbb{R}$ is a common phase velocity, the weights $w_{i,j}$ represent the coupling strength between phases $\theta_i(\tau)$ and $\theta_j(\tau)$, and $\phi_{i,j}$ is their phase difference (typically $\phi_{i,j} = -\phi_{ji}$). In traditional robotic applications that use CPGs, the phase θ is used to generate reference trajectories for the "limbs" of the robot via a parameterized map g:

$$q_{\text{ref}}(\tau) = g(p, \theta(\tau)), \tag{5.3}$$

where $q_{\text{ref}}(\tau)$ represents the reference trajectories of each actuator at time τ, and p is a set of parameters that modulate the shape of the resulting phase curves into a physical motion in space. The desired reference trajectory q_{ref} is then fed into a tracking controller, or a reference vector field (as a function of $\theta(\tau)$) that can be pushed back through g (if g is differentiable [60]). Equation (5.1) corresponds to subsystem 1 in Figure 5.1, while equation (5.3) corresponds to subsystem 2.

Designing gaits in the CPG framework is accomplished by choosing the values for parameters $w_{i,j}$, $\phi_{i,j}$, and p. Despite the widespread use of CPGs and their straightforward

implementation, there are some disadvantages to this approach that should be considered. First, it is necessary to continuously solve the differential equation (5.1) in real-time. Many approaches have been taken, including dedicated analog CPG implementations (see references within [48]). Second, the transient behavior of (5.1) may be difficult to describe. Especially in the case of gait transitions or variable velocity, since changing parameters in dynamical systems typically results in bifurcations. Such behavior can be difficult to analyze but with the approach proposed in this chapter theories of max-plus algebra can be used to analyze that behavior.

5.2.2 Buehler clock

As was mentioned in the introduction an alternative approach to CPGs for the synchronization of cyclic systems is called the "Buehler clock" [80]. The Buehler clock is illustrated in Figure 5.2 for a hexapod robot.

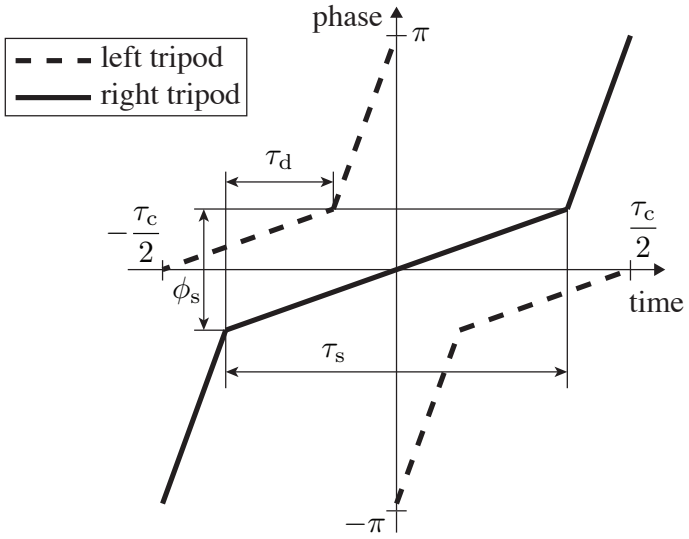

Figure 5.2: The "Buehler clock" model for a hexapod robot: piecewise constant phase velocity (Figure reproduced from Saranli et al. [80]). Each trajectory corresponds to the reference phase of a group of legs in time.

The control structure in this framework is built up as shown in Figure 5.1. The gait reference generator generates piecewise constant phase velocity reference signals. The phase velocity reference signals are based on the set of these parameters:

the cycle time is τ_c,

the stance time is τ_s,

the "stance phase" is ϕ_s,

the double stance time is τ_d, with $\tau_d = \tau_s - \tau_c/2$.

The stance phase ϕ_s represents the part of the abstracted phase when the legs are assumed to be in stance. For a gait where the legs are divided into two groups the mathematical model can be written as:

$$\theta_1(\tau) = \begin{cases} \dfrac{\phi_s}{\tau_s}\,\bar{\tau} & \text{if } -\dfrac{\tau_s}{2} < \bar{\tau} < \dfrac{\tau_s}{2}, \\ \dfrac{\pi-\phi_s}{\tau_c-\tau_s}\left(\bar{\tau}-\dfrac{\tau_s}{2}\right)+\dfrac{\phi_s}{2} & \text{if } \bar{\tau} \geq \dfrac{\tau_s}{2}, \\ \dfrac{\pi-\phi_s}{\tau_c-\tau_s}\left(\bar{\tau}+\dfrac{\tau_s}{2}\right)-\dfrac{\phi_s}{2} & \text{if } \bar{\tau} \leq -\dfrac{\tau_s}{2}, \end{cases} \tag{5.4}$$

and

$$\theta_2(\tau) = \theta_1\left(\tau+\frac{\tau_c}{2}\right), \tag{5.5}$$

with $\bar{\tau} = ((\tau+\tau_c/2) \text{ modulo } \tau_c) - \tau_c/2$. The reference phases $\theta_1(\tau)$ and $\theta_2(\tau)$ in (5.4) and (5.5), represent the right and left tripod of a hexapod robot respectively, as in Figure 5.2, and τ represents the current time instant. In the paper of Saranli et al. [80], $\theta_1(\tau)$ is used as a phase reference for legs 1, 4, and 5; and $\theta_2(\tau)$ is used for legs 2, 3, and 6, following the notation of the left-most image in Figure 5.3. The max-plus approach that is proposed in the next part of this chapter generalizes the Buehler clock approach and with the use of max-plus theories the transient and steady-state behavior is analyzed.

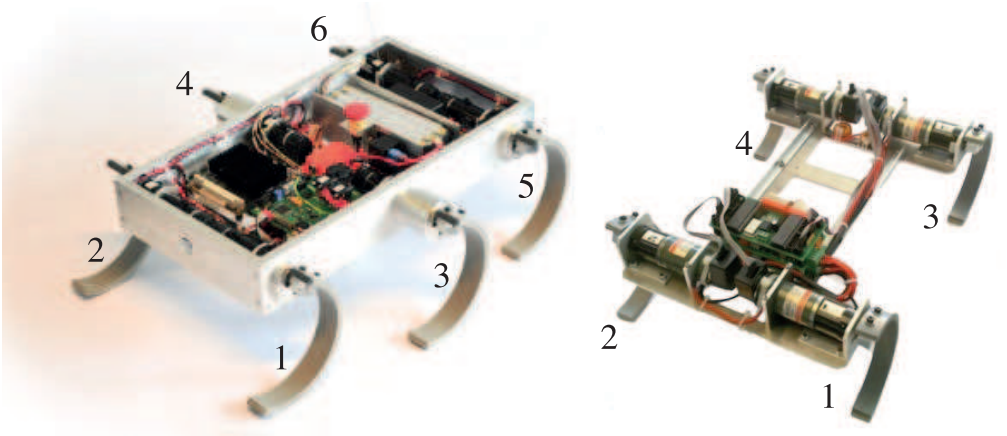

Figure 5.3: Left: Zebro and RQuad robots developed at the Delft Center for Systems and Control, morphologically identical to RHex [80]. The numbers indicate the leg labeling for each robot.

5.2.3 Switching max-plus-linear models

The timing of the touchdown and lift-off and synchronization of the legs of a robot can be described as a discrete event system. The state variables of this system are:

- $t_i(k)$ is the time instant the foot of leg i touches down for the kth cycle

- $l_i(k)$ is the time instant the foot of leg i lifts off the ground for the kth cycle

The equations describing the time evolution of the state variables and the synchronization of a walking bipedal robot are then:

$$t_1(k) = l_1(k) + \tau_{\mathrm{f}} \tag{5.6}$$
$$t_2(k) = l_2(k) + \tau_{\mathrm{f}} \tag{5.7}$$
$$l_1(k) = \max\left(t_1(k-1) + \tau_{\mathrm{g}}, t_2(k-1) + \tau_{\Delta}\right) \tag{5.8}$$
$$l_2(k) = \max\left(t_2(k-1) + \tau_{\mathrm{g}}, t_1(k) + \tau_{\Delta}\right). \tag{5.9}$$

Equations (5.6)–(5.9) capture the synchronization requirements of the legs for a traditional biped walk. Equation (5.6) states that foot 1 touches down τ_{f} time units after it has lifted off the ground. Equation (5.8) states that foot 1 will lift off the ground after both feet have spent a total of τ_{g} time units in stance and τ_{Δ} time units after foot 2 has touched down. Equations (5.7) and (5.9) have an analogous interpretation. Note that the time parameters τ_{f}, τ_{g}, and τ_{Δ} represent the *minimal* swing, stance, and double-stance times, respectively, as opposed to their exact times. Equations (5.6)–(5.9) contain only the max and + operations, motivating the use of the max-plus algebra due to the following reasons: first, (5.6)–(5.9) is nonlinear in the traditional algebra, but it is linear in the max-plus algebra; second, the theory of the max-plus algebra is well developed, and as such many properties can be inferred from the system matrices of the max-plus linear system. In the next section we explore these properties.

Equations (5.6)–(5.9) describe the synchronization constraints between two legs. These equations can be written in max-plus-linear algebra as

$$t_i(k) = l_i(k) \otimes \tau_{\mathrm{f}} \quad \text{for } i = 1,2 \tag{5.10}$$
$$l_1(k) = t_1(k-1) \otimes \tau_{\mathrm{g}} \oplus t_2(k) \otimes \tau_{\Delta} \tag{5.11}$$
$$l_2(k) = t_2(k-1) \otimes \tau_{\mathrm{g}} \oplus t_1(k-1) \otimes \tau_{\Delta}. \tag{5.12}$$

This can be extended to robots with any number of legs and for many different gaits. In order to do so we must first define for which gaits this is possible.

Definition 5.1 Leg partition
Let n be the number of legs of the robot and define m as the number of leg groups, where a leg group is defined as a set containing the indices of all legs that lift off and touch down at the same time. Let $\ell_1, \ldots, \ell_m$ be ordered sets of integers such that

$$\bigcup_{p=1}^{m} \ell_p = \{1, \ldots, n\},\ \forall i \neq j, \ell_i \cap \ell_j = \emptyset,\ \text{and}\ \forall i, \ell_i \neq \emptyset, \tag{5.13}$$

i.e., the sets ℓ_p, $p = 1, \ldots, m$, form a partition of $\{1, \ldots, n\}$. □

Definition 5.2 Gait and gait space
A gait G is defined as an ordering relation of groups of legs:

$$\mathrm{G} = \ell_1 \prec \ell_2 \prec \cdots \prec \ell_m. \tag{5.14}$$

The *gait space* is the set of all gaits that satisfy the previous definitions. □

By considering that each ℓ_p contains the indices of a set of legs that are synchronized in phase, the previous ordering relation is interpreted in the following manner: the set of legs indexed by ℓ_i swings synchronously. Once all legs in ℓ_i touch down and have been on the ground for at least τ_Δ time units, then all legs in ℓ_{i+1} initiate their swing motion. The same is true for ℓ_m and ℓ_1, closing the cycle. For example, a trotting gait, where diagonal pairs of legs move synchronously, for a quadruped robot as illustrated in Figure 5.3, is represented by:

$$\mathrm{G}_{\mathrm{trot}} = \{1,4\} \prec \{2,3\}. \tag{5.15}$$

The gait space defined above can represent gaits for which all legs have the same cycle time. As such, gaits where one leg cycles twice while another cycles only once are not captured by this model. Examples of such gaits are not common, but have been used on hexapod robots to transverse very inclined slopes sideways [93].

For an n-legged robot any gait in the gait space defined before can be described in max-plus algebra. First we can generalize equations (5.10), (5.11), and (5.12) for n-legged robots by defining the following vectors:

$$t(k) = [t_1(k) \ \cdots \ t_n(k)]^\top \tag{5.16}$$
$$l(k) = [l_1(k) \ \cdots \ l_n(k)]^\top. \tag{5.17}$$

Equation (5.10) is then written as:

$$t(k) = \tau_{\mathrm{f}} \otimes l(k). \tag{5.18}$$

If one assumes that the synchronization is always enforced on the lift-off time of a leg, equations (5.11) and (5.12) are written jointly as:

$$l(k) = \tau_{\mathrm{g}} \otimes t(k-1) \oplus P \otimes t(k) \oplus Q \otimes t(k-1), \tag{5.19}$$

where the matrices P and Q encode the synchronization between lift-off of a leg related to a touchdown of the current event (as in equation (5.9)) and a touchdown of the previous event (as in equation (5.8)), respectively. The rationale behind this particular model is to prevent that a legged platform has too many legs in swing while walking[3], and the robot risks falling down. Synchronization constraints are always imposed on legs that are in stance and are about to swing: some legs should only swing if others are in stance (equation (5.19)). By doing so we can ensure that for any disturbance at most one group of legs is in swing and therefore we can guarantee that a minimum number of legs is always on the ground. This ensures kinematic stance stability, under the assumption that the gait during steady state is kinematic stance-stable, i.e. no matter which group of legs is in swing, the legs in the other groups ensure the robot is kinematic stance-stable. Once in swing, legs are never constrained to go into stance (equation (5.18)). With the

[3]As mentioned previously in this chapter we do not consider gaits with aerial phases, although it can still be achieved using the same class of models.

previously defined gait notation matrices P and Q can be derived:

$$[P]_{pq} = \begin{cases} \tau_\Delta & \forall j \in \{1,\ldots,m-1\}; \forall p \in \ell_{j+1}; \forall q \in \ell_j \\ \varepsilon & \text{otherwise} \end{cases} \tag{5.20}$$

$$[Q]_{pq} = \begin{cases} \tau_\Delta & \forall p \in \ell_1; \forall q \in \ell_m \\ \varepsilon & \text{otherwise} \end{cases} . \tag{5.21}$$

Matrix P ensures that the lift-off of the legs in ℓ_{j+1} in the current cycle is τ_Δ time units after the touchdown of the legs in ℓ_j in the current cycle (for all $j \in \{1,\ldots,m-1\}$). Matrix Q ensures that the lift-off of the legs in ℓ_1 in the current cycle is after the touchdown of ℓ_m in the previous cycle. These matrices ensure that the synchronization between the sets of legs and ensure only one set of legs can be in the air at all times, even during disturbances.

For the trotting gait $\mathrm{G}_{\mathrm{trot}}$ we obtain:

$$P_{\mathrm{trot}} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_\Delta & \varepsilon & \varepsilon & \tau_\Delta \\ \tau_\Delta & \varepsilon & \varepsilon & \tau_\Delta \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \quad \text{and} \quad Q_{\mathrm{trot}} = \begin{bmatrix} \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon \end{bmatrix}. \tag{5.22}$$

Equations (5.18) - (5.22) can be written in state-space form as:

$$\begin{bmatrix} t(k) \\ l(k) \end{bmatrix} = \left[\begin{array}{c|c} \mathcal{E} & \tau_{\mathrm{f}} \otimes E \\ \hline P & \mathcal{E} \end{array}\right] \otimes \begin{bmatrix} t(k) \\ l(k) \end{bmatrix} \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline \tau_{\mathrm{g}} \otimes E \oplus Q & E \end{array}\right] \otimes \begin{bmatrix} t(k-1) \\ l(k-1) \end{bmatrix}. \tag{5.23}$$

Define the matrices

$$A_0 = \left[\begin{array}{c|c} \mathcal{E} & \tau_{\mathrm{f}} \otimes E \\ \hline P & \mathcal{E} \end{array}\right]; \qquad A_1 = \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline \tau_{\mathrm{g}} \otimes E \oplus Q & E \end{array}\right]. \tag{5.24}$$

Consider the full state x defined as $x(k) = [t^\top(k)\ l^\top(k)]^T$. Equation (5.23) can then be written in simplified notation:

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1). \tag{5.25}$$

Note that additional max-plus identity matrices E are introduced in the diagonal of matrix A_1. This results in the extra trivial constraints $t_i(k) \geq t_i(k-1)$ and $l_i(k) \geq l_i(k-1)$, also resulting in the final system matrix (defined in page 112, equation (5.35)) being irreducible. This is observed later on in Lemma 5.4.

Define the function $\flat$[4] that transforms a gait into a vector of integers:

$$\begin{aligned} \flat \ : \ & \{[\ell_1]_1,\ldots,[\ell_1]_{i_1}\} \prec \cdots \prec \{[\ell_m]_1,\ldots,[\ell_m]_{i_m}\} \mapsto \\ & [[\ell_1]_1,\ldots,[\ell_1]_{i_1} \ldots [\ell_m]_1,\ldots,[\ell_m]_{i_m}]^\top . \end{aligned} \tag{5.26}$$

Using again the previous trotting example we get that $\flat(\mathrm{G}_{\mathrm{trot}}) = [1\ 4\ 2\ 3]^\top$. Note that the gaits $\{1,4\} \prec \{2,3\}$ and $\{4,1\} \prec \{2,3\}$ although resulting in indistinguishable motion in practice, have different mathematical representations since
$\flat(\{1,4\} \prec \{2,3\}) \neq \flat(\{4,1\} \prec \{2,3\})$.

[4]The symbol *flat* "$\flat$" is chosen since it "flattens" the ordered collection of ordered sets of a gait into a vector.

Definition 5.3 Normal gait
A gait $\bar{\mathrm{G}}$ is called a *normal gait* if the elements of the vector $\flat\left(\bar{\mathrm{G}}\right)$ are in increasing numerical order. □

For a gait G, define the similarity matrix $\bar{C} \in \mathbb{R}_{\varepsilon}^{n\times n}$ as:

$$\left[\bar{C}\right]_{i,j} = \begin{cases} e & \text{if } [\flat(\mathrm{G})]_i = j \\ \varepsilon & \text{otherwise} \end{cases}, \forall i,j \in \{1,\ldots,n\}. \tag{5.27}$$

The similarity matrix $\bar{C}$ is such that $\bar{C} \otimes \bar{C}^{\top} = \bar{C}^{\top} \otimes \bar{C} = E$.

The similarity matrix associated with the trotting gait $\mathrm{G}_{\mathrm{trot}}$ is:

$$\bar{C}_{\mathrm{trot}} = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon \end{bmatrix}. \tag{5.28}$$

The similarity matrix $\bar{C}$ has the property of "normalizing" the P and Q matrices to a max-plus algebraic lower triangular form $\bar{P}$ and a max-plus algebraic upper triangular form $\bar{Q}$ respectively:

$$\bar{P} = \bar{C} \otimes P \otimes \bar{C}^{\top} \tag{5.29}$$

$$\bar{Q} = \bar{C} \otimes Q \otimes \bar{C}^{\top}. \tag{5.30}$$

Taking the previous example of the trotting gait, the normalized matrices take the form

$$\bar{P}_{\mathrm{trot}} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_\Delta & \tau_\Delta & \varepsilon & \varepsilon \\ \tau_\Delta & \tau_\Delta & \varepsilon & \varepsilon \end{bmatrix} \text{ and } \bar{Q}_{\mathrm{trot}} = \begin{bmatrix} \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta \\ \varepsilon & \varepsilon & \tau_\Delta & \tau_\Delta \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \tag{5.31}$$

which are generated by the normal gait $\{1,2\} \prec \{3,4\}$. Let $\#\ell_i$ represent the number of elements of the set ℓ_i and define $\mathbb{1}_{\#\ell_i \times \#\ell_j}$ as a matrix of dimensions $\#\ell_i \times \#\ell_j$ containing only ones. Then for a general normal gait

$$\bar{\mathrm{G}} = \ell_1 \prec \ell_2 \prec \cdots \prec \ell_m, \tag{5.32}$$

with $\bar{\mathbb{1}}_{i,j} = \mathbb{1}_{\#\ell_i \times \#\ell_j}$ the structure of $\bar{P}$ and $\bar{Q}$ is:

$$\bar{P} = \begin{bmatrix} \mathcal{E} & & & \cdots & \mathcal{E} \\ \tau_\Delta \otimes \bar{\mathbb{1}}_{2,1} & \mathcal{E} & & & \vdots \\ \mathcal{E} & \tau_\Delta \otimes \bar{\mathbb{1}}_{3,2} & \mathcal{E} & & \\ \vdots & & \ddots & & \\ \mathcal{E} & \cdots & & \tau_\Delta \otimes \bar{\mathbb{1}}_{m,m-1} & \mathcal{E} \end{bmatrix}, \tag{5.33}$$

and

$$\bar{Q} = \begin{bmatrix} \mathcal{E} & \tau_\Delta \otimes \bar{\mathbb{1}}_{1,m} \\ \mathcal{E} & \mathcal{E} \end{bmatrix}. \tag{5.34}$$

From (5.33) it is clear that the matrix $\bar{P}$ is always max-plus nilpotent, since the upper triangle is max-plus zero.

Lemma 5.1
Max-plus nil-potency is invariant to max-plus similarity transformations (e.g. as defined in equations (5.29), (5.30)).

Proof: See Lopes et al. [68]. □

Given an arbitrary gait G with associated matrices P, Q, A_0, and A_1 one can find the normal matrix $\bar{P}$ which is max-plus nilpotent. From Lemma 5.1 P is then also max-plus nilpotent.

Lemma 5.2
A sufficient condition for A_0^* to exist is that the matrix P is nilpotent in the max-plus sense.

Proof: See Lopes et al. [68]. □

Since P is always max-plus nilpotent for gaits generated by expressions (5.20) and (5.21), we conclude that A_0^* is well defined. In the beginning of this section we have presented the implicit form of the synchronization equations (5.25). However, if A_0^* exists then using equations (2.6) and (2.7), system (5.25) can be transformed into an explicit set of equations.

Let A, which we call *system matrix*, be defined by:

$$A = A_0^* \otimes A_1. \tag{5.35}$$

Using Theorem 2.1 equation (5.25) can be rewritten as:

$$x(k) = A_0^* \otimes A_1 \otimes x(k-1) = A \otimes x(k-1). \tag{5.36}$$

With the system matrix A defined the max-plus theory from Chapter 2 can be used to analyze the behavior of the system described in (5.36).

For the analysis of the system the system matrix A is transformed using a similarity transformation with matrix:

$$C = \begin{bmatrix} \bar{C} & \mathcal{E} \\ \mathcal{E} & \bar{C} \end{bmatrix}. \tag{5.37}$$

The similarity matrix C transforms the system matrix A of an arbitrary gait G into the system matrix $\bar{A}$ of a normal gait $\bar{\text{G}}$ via the similarity transformation $\bar{A} = C \otimes A \otimes C^{\top}$.

The structure of $\bar{A}$ is derived and analyzed in Lopes et al. [68] and (5.38)–(5.40) illustrate the resulting structure of $\bar{A}$ written in the system form $\bar{x}(k) = \bar{A} \otimes \bar{x}(k-1)$, with $\bar{x}(k) = C \otimes x(k)$, and $\bar{E}_i = E_{\#\ell_i}$.

$$\bar{x}(k) = \bar{A} \otimes \bar{x}(k-1) \Leftrightarrow \tag{5.38}$$

$$\bar{x}(k) = \left[\begin{array}{c||c} \tau_{\mathrm{f}} \otimes (\tau_{\mathrm{g}} \otimes W \oplus V) & \tau_{\mathrm{f}} \otimes W \\ \hline\hline \tau_{\mathrm{g}} \otimes W \oplus V & W \end{array}\right] \otimes \bar{x}(k-1) \Leftrightarrow \tag{5.39}$$

$$\underbrace{\left[\begin{array}{c} t_{\ell_1}(k) \\ \vdots \\ \hline\hline t_{\ell_m}(k) \\ \hline\hline l_{\ell_1}(k) \\ \vdots \\ \hline l_{\ell_m}(k) \end{array}\right]}_{\bar{x}(k)} = \underbrace{\left[\begin{array}{c|c||c} A_{11} & A_{12} & A_{13} \\ \hline\hline A_{21} & A_{22} & A_{23} \\ \hline\hline A_{31} & A_{32} & A_{33} \\ \hline A_{41} & A_{42} & A_{43} \end{array}\right]}_{\bar{A}} \otimes \underbrace{\left[\begin{array}{c} t_{\ell_1}(k-1) \\ \vdots \\ \hline\hline t_{\ell_m}(k-1) \\ \hline\hline l_{\ell_1}(k-1) \\ \vdots \\ \hline l_{\ell_m}(k-1) \end{array}\right]}_{\bar{x}(k-1)}, \tag{5.40}$$

with (the $\otimes$ operator is omitted in unambiguous locations):

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] = \left[\begin{array}{ccc|c} \tau_\gamma \bar{E}_1 & \cdots & \mathcal{E} & \tau_\delta \bar{\mathbb{1}}_{1,m} \\ \tau_\gamma \tau_\delta \bar{\mathbb{1}}_{2,1} & \tau_\gamma \bar{E}_2 & \vdots & \tau_\delta^{\otimes 2} \bar{\mathbb{1}}_{2,m} \\ \vdots & & \ddots & \vdots \\ \hline \tau_\gamma \tau_\delta^{\otimes(m-1)} \bar{\mathbb{1}}_{m,1} & \cdots & \tau_\gamma \tau_\delta \bar{\mathbb{1}}_{m,m-1} & \tau_\gamma \bar{E}_m \oplus \tau_\delta^{\otimes m} \bar{\mathbb{1}}_{m,m} \end{array}\right] \tag{5.41}$$

$$\left[\begin{array}{c|c} A_{31} & A_{32} \\ \hline A_{41} & A_{42} \end{array}\right] = \left[\begin{array}{ccc|c} \tau_{\mathrm{g}} \bar{E}_1 & \cdots & \mathcal{E} & \tau_\Delta \bar{\mathbb{1}}_{1,m} \\ \tau_{\mathrm{g}} \tau_\delta \bar{\mathbb{1}}_{2,1} & \tau_{\mathrm{g}} \bar{E}_2 & \vdots & \tau_\Delta \tau_\delta \bar{\mathbb{1}}_{2,m} \\ \vdots & & \ddots & \vdots \\ \hline \tau_{\mathrm{g}} \tau_\delta^{\otimes(m-1)} \bar{\mathbb{1}}_{m,1} & \cdots & \tau_{\mathrm{g}} \tau_\delta \bar{\mathbb{1}}_{m,m-1} & \begin{array}{c} \tau_{\mathrm{g}} \bar{E}_m \oplus \\ \tau_\Delta \tau_\delta^{\otimes(m-1)} \bar{\mathbb{1}}_{m,m} \end{array} \end{array}\right] \tag{5.42}$$

$$\left[\begin{array}{c} A_{13} \\ \hline\hline A_{23} \\ \hline\hline A_{33} \\ \hline A_{43} \end{array}\right] = \left[\begin{array}{ccc} \tau_{\mathrm{f}} \bar{E}_1 & \cdots & \mathcal{E} \\ \tau_{\mathrm{f}} \tau_\delta \bar{\mathbb{1}}_{2,1} & \tau_{\mathrm{f}} \bar{E}_2 & \vdots \\ \vdots & \ddots & \\ \hline\hline \tau_{\mathrm{f}} \tau_\delta^{\otimes(m-1)} \bar{\mathbb{1}}_{m,1} & \cdots & \tau_{\mathrm{f}} \bar{E}_m \\ \hline\hline \bar{E}_1 & \cdots & \mathcal{E} \\ \tau_\delta \bar{\mathbb{1}}_{2,1} & \bar{E}_2 & \vdots \\ \vdots & \ddots & \\ \hline \tau_\delta^{\otimes(m-1)} \bar{\mathbb{1}}_{m,1} & \cdots & \bar{E}_m \end{array}\right]. \tag{5.43}$$

5.2.4 Control structure

In this section the modular control structure is given that implements the presented max-plus framework both in simulation and in reality on the legged robots illustrated in Figure 5.3. This structure, illustrated in Figure 5.4, consists of four control blocks: the supervisory controller, tasked with choosing a gait; the max-plus gait generator, which generates an event schedule; the continuous time scheduler, which transforms the event schedule into a continuous time reference trajectory; and finally the tracking controller,

which enforces the desired reference trajectory. Note that both the supervisory controller and the max-plus gait generator blocks use feedback on the phase state to update the internal scheduling.

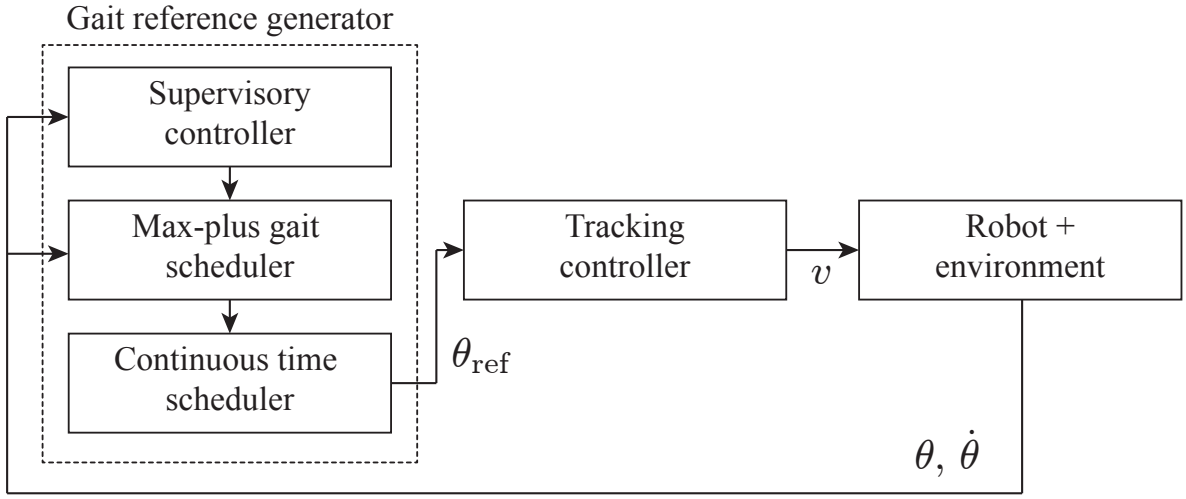

Figure 5.4: Block diagram of the control structure using the max-plus algebra framework for legged locomotion presented in this chapter.

The choice of the supervisory controller can be a function of the terrain, desired speed, or other considerations. Section 5.4 will be dedicated to gait transitions, further elaborating on the operation of the supervisory control block.

The event schedule $S \in \mathbb{R}_{\varepsilon}^{2n\times(2p+1)}$ for $p \geq 1$ (in the case of the robots utilized in this chapter $p = 1$ is used) is defined to be the matrix

$$S = \begin{bmatrix} x(k-p) & \cdots & x(k) & \cdots & x(k+p) \end{bmatrix}. \tag{5.44}$$

Denote the i-th row of S as $[S]_{i,\cdot}$. Then the parameters p and k are chosen such that at the time instant τ for each row of S we get that

$$\min([S]_{i,\cdot}) < \tau < \max([S]_{i,\cdot}), \tag{5.45}$$

i.e., for each leg S contains both events that have happened in the past and events that are scheduled to occur in the future (in a practical implementation the matrix S is updated at discrete time instants that are a function of the total cycle time, thus S is actually a function of time). If we consider that foot i always touches down when its phase is at a fixed value θ_{t} and always lifts off the ground at the fixed phase θ_{l}, then it is possible to generate a reference phase via the function

$$\theta_{\text{ref}}(\tau, S(\tau)) : \mathbb{R} \times \mathbb{R}_{\varepsilon}^{2n\times(2p+1)} \to \mathbb{T}^n, \tag{5.46}$$

that takes as inputs time $\tau \in \mathbb{R}$ and the event schedule and outputs a piecewise affine

trajectory for each of the leg's phases:

$$[\theta_{\text{ref}}]_i := \begin{cases} \dfrac{\theta_{\text{l}}\,(t_i(k_{\text{t}i}) - \tau) + (\theta_{\text{t}} + 2\pi)\,(\tau - l_i(k_{\text{l}i}))}{t_i(k_{\text{t}i}) - l_i(k_{\text{l}i})} \\ \qquad\qquad \text{if } \tau \in [l_i(k_{\text{l}i}), t_i(k_{\text{t}i})) \\ \dfrac{\theta_{\text{t}}\,(l_i(k_{\text{l}i}+1) - \tau) + \theta_{\text{l}}\,(\tau - t_i(k_{\text{t}i}))}{l_i(k_{\text{l}i}+1) - t_i(k_{\text{t}i})} \\ \qquad\qquad \text{if } \tau \in [t_i(k_{\text{t}i}), l_i(k_{\text{l}i}+1)) \end{cases} . \tag{5.47}$$

Since the legs do not all lift off and touch down at the same time, some legs may have already finished the current cycle k and have started a new cycle $k+1$, while other legs are still in cycle k, therefore we differentiate between the cycle counters for each leg with the event counter variables $k_{\text{t}i}$ and $k_{\text{l}i}$, where i indicates the index of the leg. The interpretation of expression (5.47) is as follows: the function θ_{ref} interpolates the phase parameters θ_{t} and θ_{l} linearly in time τ. For a specific leg i, if it is in stance, then the interval $[t_i(k_{\text{t}i}), l_i(k_{\text{l}i}+1))$ is used for interpolation of the phase, such that at time $\tau = t_i(k_{\text{t}i})$ the reference phase is θ_{t} and at time $\tau = l_i(k_{\text{l}i}+1)$ the reference phase for leg i is θ_{l}. If the leg is in swing, then the interpolation interval $[l_i(k_{\text{l}i}), t_i(k_{\text{t}i}))$ is used instead. Figure 5.5 illustrates a sample simulation where the reference phase is represented by the dashed lines. For each leg, each graph ranges from $-\pi$ to π and the phase "wraps around" when crossing π, as illustrated by the vertical lines.

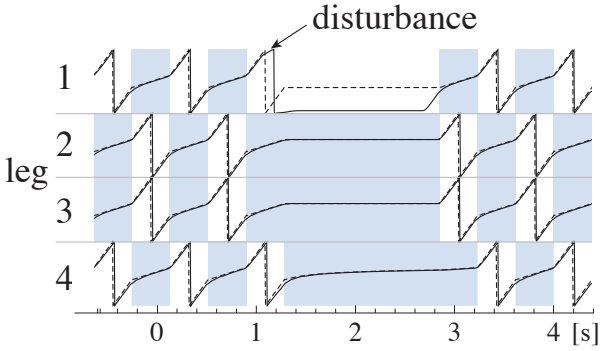

Figure 5.5: Experimental run on a quadruped: the dashed lines represent the reference phase, ranging from $(-\pi, \pi]$ (vertical lines represent the phase wrapping around), the solid lines represent the actual phase of each leg. In this experiment, leg 1 is prevented from touching down, resulting in the phase of all other legs to be delayed.

Note that function $\theta_{\text{ref}}(\tau, S(\tau))$ is general, and can be used instead of a CPG type generator, as in (5.1), resulting in a new discrete-event type of reference trajectory generator for the actuators of the robot:

$$q_{\text{ref}}(\tau) \;=\; g(p, \theta_{\text{ref}}(\tau, S(\tau))). \tag{5.48}$$

For a tripod gait $\{1,4,5\} \prec \{2,3,6\}$ with the parameters $\phi_{\text{s}} = \theta_{\text{t}} + \theta_{\text{l}}$, (5.47) results in the Buehler Clock equations (5.4)-(5.5). Thus, the switching max-plus method is a generaliza-

tion of the Buehler Clock. We can now establish a comparison between the standard CPG versus the switching max-plus methodology, illustrated in Table 5.1. CPGs use differential equations to describe continuous dynamics whereas the proposed switching max-plus approach uses max-plus-linear equations to describe a discrete event system representation of the dynamics. For CPGs the desired gait can be achieved by meticulously choosing the appropriate phase offset parameters and two gains to get an attractive limit cycle that determines the phase differences between the legs, for the switching max-plus approach the gait is determined by an ordered set of numbers and three time parameters, resulting in a max-plus-linear system matrix associated to the gait. The eigenvalue and eigenvector determine the cycle time and timing between the groups of legs. For the max-plus approach the convergence to the steady state behavior is guaranteed to occur in at most two cycles after the last disturbance has occurred while remaining kinematic stance stable. This is based on Lemma 5.8 in Section 5.3.3. For the CPG approach convergence depends on the chosen gains and might not happen without causing instability. Stable gait transitions are guaranteed for the switching max-plus approach thanks to the way the synchronization is set up. For CPGs stable gait transitions can only be achieved by encoding subspaces in the vector fields describing the phase behavior of the legs that must be avoided. To implement the CPG approach in robots differential equations must be solved on-line using numerical algorithms. For the switching max-plus approach calculation of the event times can be done with additions and maximizations and the event times must be used to design piece-wise affine reference velocity signals using interpolation.

Table 5.1: Comparison between standard CPG and switching max-plus methods.

Property	CPG	Switching max-plus
Dynamics	continuous	discrete
System representation	differential equation (5.1)	max-plus linear system (5.23)
Control parameterization	set of phase offset parameters and gains: $w_{i,j}$, $\phi_{i,j}$	ordered set of numbers $\ell_1 \prec \ell_2 \prec \cdots \prec \ell_m$ and time parameters $\tau_{\mathrm{f}}, \tau_{\mathrm{g}}, \tau_{\Delta}$
Steady state	limit cycle	max-plus eigenvector
Cycle time	depends on the gain	max-plus eigenvalue
Convergence	depends on the gain	maximum 2 cycles
Transitions with constraint guarantees	obstacles encoded in vector fields	switch state matrices
Implementation	numerical differential equation solver	additions, maximizations, linear interpolation

5.3 Max-plus eigenstructure of the system matrix

Given a parameterization of a gait, it is fundamental to understand whether the system $x(k) = A \otimes x(k-1)$ reaches a unique steady state behavior. In robotics this is the equivalent to asking: "does the legged robot move as specified?" and "If one of the legs gets blocked by an obstacle for a short time and therefore cannot move can the robot recover from this?". These questions are answered by analyzing the max-plus eigenstructure of the system matrix: a unique eigenvalue means that the legs have a unique cycle time, and a unique (up to scaling) eigenvector means that the legs always reach the same motion pattern, independently of the initial condition or temporary disturbances. The results obtained below use various analysis techniques available for max-plus linear systems. This is necessary due to the intrinsic time structure associated with the problem. In Section 5.3.2 we show that for a fixed structure (i.e. a single Petri net) unique or non-unique eigenvectors are found by changing the holding time parameters.

Consider the following assumption (which is always satisfied in practice since the leg swing and stance times are always positive numbers):

Assumption 1 (A1) $\tau_{\mathrm{g}}, \tau_{\mathrm{f}}, \tau_{\Delta} > 0$.

Furthermore let:

$$\tau_{\delta} = \tau_{\mathrm{f}} \otimes \tau_{\Delta} \text{ and } \tau_{\gamma} = \tau_{\mathrm{f}} \otimes \tau_{\mathrm{g}}. \tag{5.49}$$

Then the following lemma defines an eigenvalue and eigenvector for the system matrix A.

Lemma 5.3
If Assumption A1 is satisfied then

$$\lambda := \tau_{\delta}^{\otimes m} \oplus \tau_{\gamma}, \tag{5.50}$$

where λ is a max-plus eigenvalue of the system matrix A (and $\bar{A}$) defined by (5.36), and $v \in \mathbb{R}_{\varepsilon}^{2n}$ defined by

$$\forall j \in \{1, \ldots, m\}, \forall q \in \ell_j : \qquad [v]_q := \tau_{\mathrm{f}} \otimes \tau_{\delta}^{\otimes j-1} \tag{5.51}$$

$$[v]_{q+n} := \tau_{\delta}^{\otimes j-1}, \tag{5.52}$$

where v is a max-plus eigenvector of A.

Proof: See Lopes et al. [68] □

Consider again the trotting gait for a quadruped G_{trot} defined in (5.15). For this gait $m = 2$, resulting in:

$$v_{\text{trot}} = \begin{bmatrix} \tau_{\text{f}} \\ \tau_{\Delta} \otimes \tau_{\text{f}}^{\otimes 2} \\ \tau_{\Delta} \otimes \tau_{\text{f}}^{\otimes 2} \\ \tau_{\text{f}} \\ 0 \\ \tau_{\Delta} \otimes \tau_{\text{f}} \\ \tau_{\Delta} \otimes \tau_{\text{f}} \\ 0 \end{bmatrix}; \qquad \lambda_{\text{trot}} = (\tau_{\text{f}} \otimes \tau_{\Delta})^{\otimes 2} \oplus \tau_{\text{f}} \otimes \tau_{\text{g}}. \tag{5.53}$$

Lemma 5.4
The system matrix A is irreducible.

Proof: See Lopes et al. [68]. □

Corollary 5.5 *The max-plus eigenvalue λ of A is given by (5.50) is unique.*

The max-plus eigenvector v defined by (5.51)–(5.52) is not necessarily unique, given Assumption A1 alone. Since, to the author's best knowledge, there exists no algebraic method to prove uniqueness of a max-plus eigenvector in general, without having to determine the critical graph first, we take advantage of the critical graph of A to further investigate this property. If the critical graph of an irreducible max-plus system matrix has a single strongly connected subgraph, then its max-plus eigenvector is unique up to a max-plus scaling factor (see Theorem 3.101 in the work of Baccelli et al. [3],). We proceed by computing the precedence graph of $\bar{A}$.

5.3.1 Precedence graph of $\bar{A}$

Since this graph can be quite large for a general $\bar{A}$, it is more efficient to first group "similar" nodes into a single node, i.e. apply a procedure called node reduction (Figure 5.6).

Next various subgraphs of the graph of $\bar{A}$ are shown to better illustrate its structure (Figure 5.7). The total precedence graph of $\bar{A}$ is thus the combination of figures of the form 5.6 and 5.7. The process of constructing the graph of $\bar{A}$ starts by grouping all nodes of an event associated with a group of legs ℓ_i into a single node. This can be accomplished since the incoming and outgoing arcs of event nodes from the same group of legs ℓ_i for the most part connect to the same nodes. As an example, consider the first set of $\#\ell_1$ rows of $\bar{A}$ as defined in expression (5.40):

$$\begin{aligned} t_{\ell_1}(k) \;=\; & \tau_{\gamma} \otimes E_1 \otimes t_{\ell_1}(k-1) \oplus \tau_{\delta} \otimes \mathbb{1}_{1,m} \otimes t_{\ell_m}(k-1) \oplus \\ & \tau_{\text{f}} \otimes E_1 \otimes l_{\ell_1}(k-1). \end{aligned} \tag{5.54}$$

The precedence graph for equation (5.54) consists of $3 \times \#\ell_1$ nodes, since it involves the vectors t_{ℓ_1}, t_{ℓ_m}, and l_{ℓ_1}. The relation between $t_{\ell_1}(k)$ and $t_{\ell_1}(k-1)$ in this example results

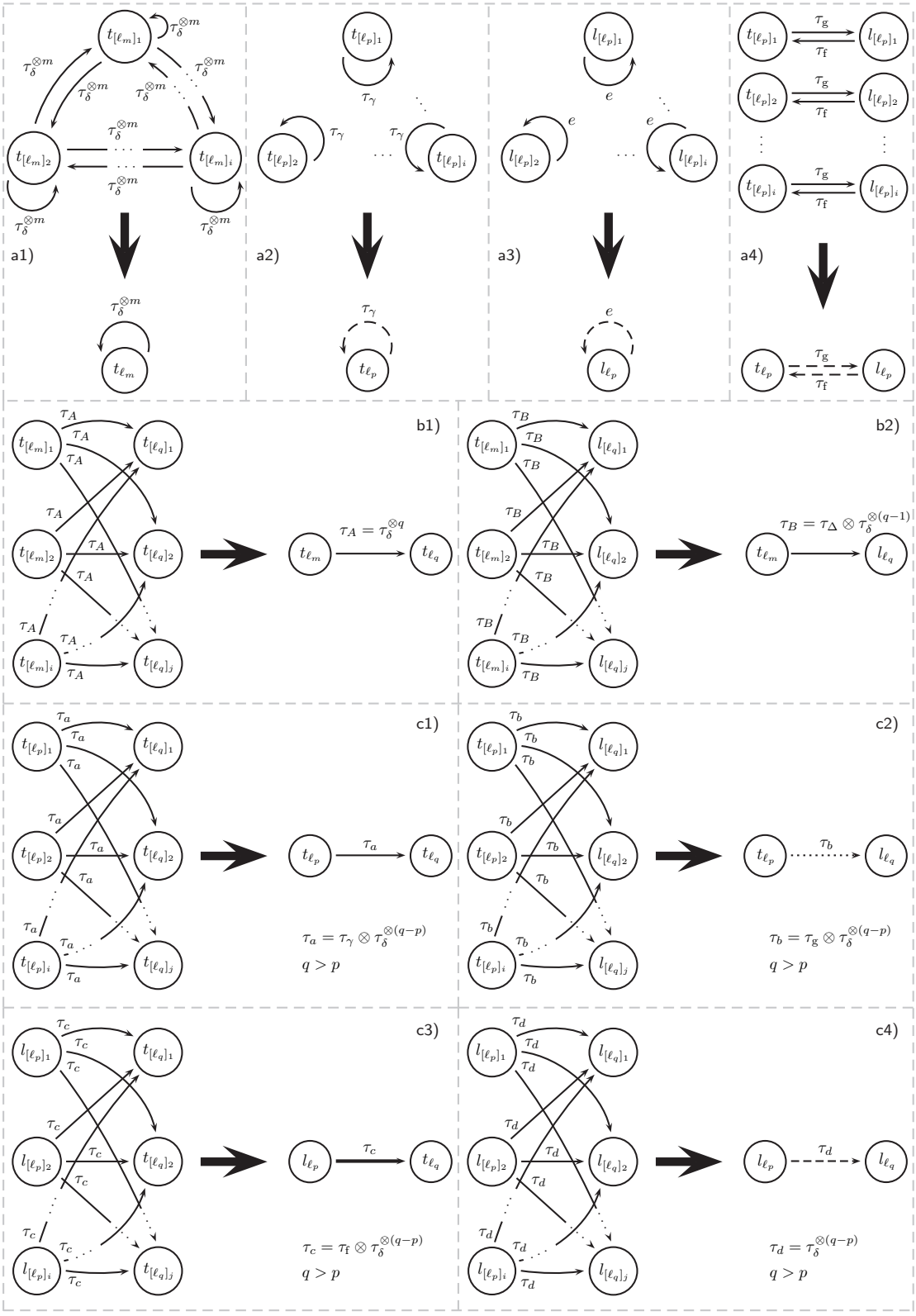

Figure 5.6: Graph reductions. Touchdown and lift off events with indices belonging to the same set ℓ_q can be grouped together since they have the same number of output and input arcs with the same weights.

in $\#\ell_1$ self-connected arcs in the t_{ℓ_1} events with weights τ_γ. Instead of expressing all elements of t_{ℓ_1} as individual nodes with self-arcs, we reduce then to a single node with one self-arc, as seen in Figure 5.6-a2. The dashed attribute used on the self-arc indicates that for each node in the group only self-arcs exist, as expressed by the matrix E_1. The relation between $t_{\ell_1}(k)$ and $t_{\ell_m}(k-1)$ is somewhat more involved, since it contains $\#\ell_1 \times \#\ell_m$ arcs, as expressed by the matrix $\mathbb{1}_{1,m}$. The resulting node reduction is illustrated in Figure 5.6-b1. The node reduction for the relation between t_{ℓ_1} and l_{ℓ_1} is illustrated in Figure 5.6-a4. Again we use dashed attributes on the arcs to represent the connecting matrix E_1. For all other relations with connecting matrices $\mathbb{1}$ we use solid arcs. We make an exception in Figures 5.6-c1 to 5.6-c4 where different line attributes are used to distinguish arcs from $t_{\ell_p} \to t_{\ell_q}$, $t_{\ell_p} \to l_{\ell_q}$, etc. The same line attributes are used in Figures 5.7-c1 and 5.7-c2. Note that multiple incoming arcs to a node are related via the $\oplus$ operation, e.g. as in the example (5.54) the node t_{ℓ_1} has 3 incoming arcs, illustrated in Figure 5.7.

The following list summarizes the node reduction:

- Figure 5.6-a1 illustrates node reduction of the term $\tau_\delta^{\otimes m} \otimes \mathbb{1}_{m,m}$ of the sub matrix A_{22} from expression (5.41).
- Figure 5.6-a2 illustrates the node reduction of the block diagonal of the sub matrix A_{11} and the $\tau_\gamma \otimes \bar{E}_m$ term of A_{22}.
- Figure 5.6-a3 illustrates the node reduction of the block diagonal of the sub matrix $[A_{33}^\top \;\; A_{43}^\top]^\top$.
- Figure 5.6-a4 illustrates the node reduction of the term $\tau_\mathrm{g} \otimes E_m$ of the sub matrix A_{42} together with the block diagonals of matrices A_{31} and $[A_{13}^\top \;\; A_{23}^\top]^\top$.
- Figures 5.6-b1 and 5.6-b2 illustrate the node reduction for the columns formed by the sub matrices A_{12} and $[A_{32}^\top \;\; A_{42}^\top]^\top$ respectively (not including the term $\tau_\mathrm{g} \otimes E_m$ from matrix A_{42} already represented in Figure 5.6-a4).
- Figures 5.6-c1 to 5.6-c4 illustrate the node reduction of the off-diagonal elements of the matrices $\tau_\gamma \otimes W$, $\tau_\mathrm{f} \otimes W$, $\tau_\mathrm{g} \otimes W$, and W, from expression (5.39) respectively.

Given the node reduction one can now proceed to construct the precedence graph of $\bar{A}$:

- Figure 5.7-a is the graph of the block diagonal of $\bar{A}$ together with the block diagonals of the sub matrices $\begin{bmatrix} A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix}$ and $[A_{13}^\top \;\; A_{23}^\top]^T$ using the node reductions presented in Figures 5.6-a1 to 5.6-a4.
- Figure 5.7-b is the graph of the columns formed by the sub matrices A_{12} and $[A_{32}^\top \;\; A_{42}^\top]^\top$ using node reductions presented in Figures 5.6-b1 and 5.6-b2.
- Figures 5.7-c1 and 5.7-c2 illustrate two subgraphs of the remaining columns of $\bar{A}$. Note that we only present the subgraphs of the first sets of $\#\ell_1$ and $\#\ell_2$ out of a total of $m-1$ columns. These follow the same pattern. We use different attributes on the arcs, such as dashed, thick solid, etc., to distinguish the different node reductions, as presented in Figures 5.6-c1 to 5.6-c4.

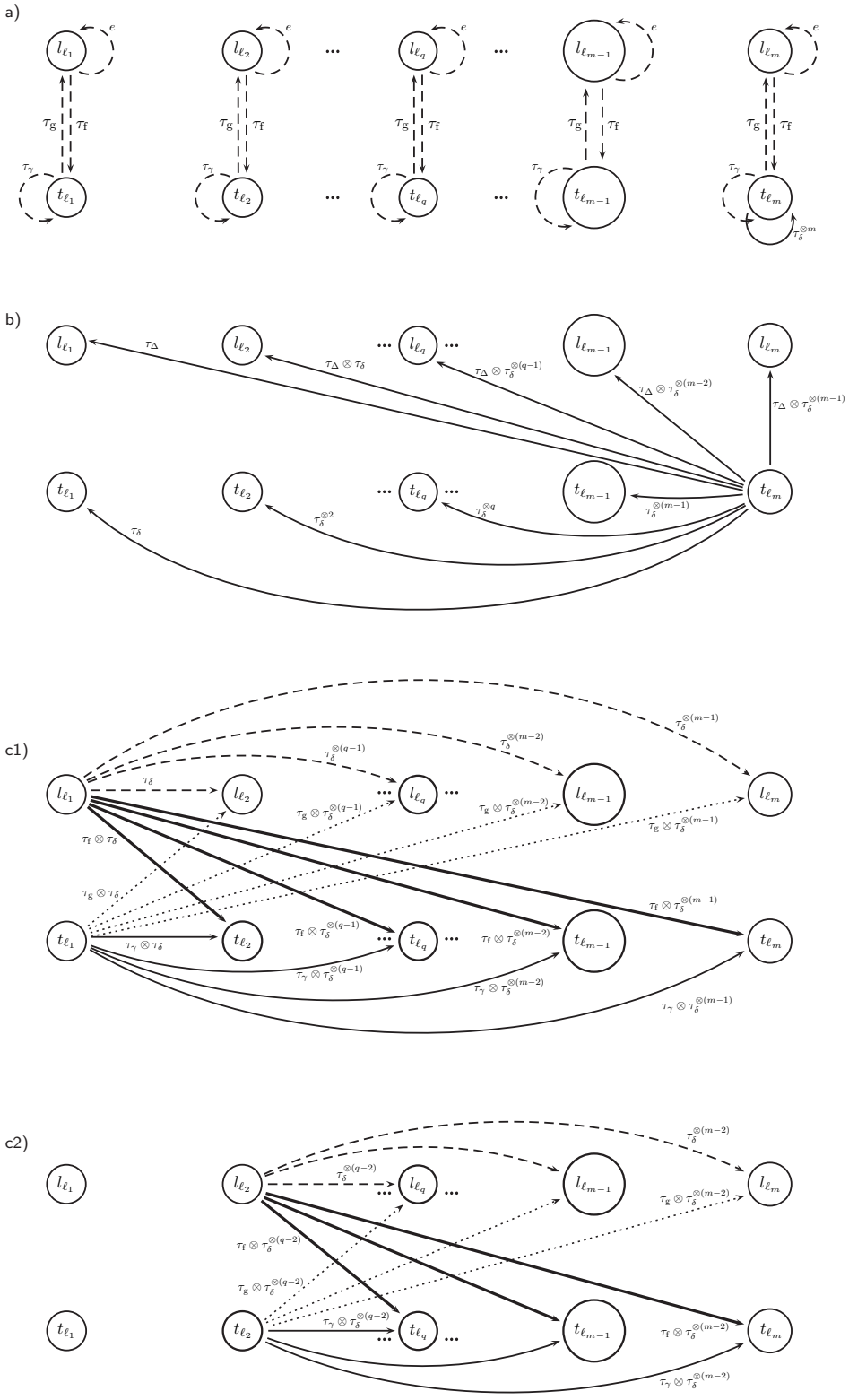

Figure 5.7: Elements of the precedence graph of the system matrix A. The total precedence graph of A is composed of all the arcs presented in a) and b), together with the $m-1$ remaining subgraphs that follow the pattern of Figures c1 and c2.

5.3.2 The critical graph of A

Before we look at the critical graph of A we first use a similarity transformation on A to get $\bar{A}$ and then we derive the critical graph of $\bar{A}$ from the precedence graph of $\bar{A}$. This is done in the proof of Lemma 5.6. Recall that the *critical graph* $\mathcal{G}_c(A)$ of the matrix A is the set of all critical circuits and that every circuit of the precedence graph $\mathcal{G}(A)$ with an average weight that is equal to $\lambda_{\max}$ is called a critical circuit. The critical graphs of A and $\bar{A}$ are equivalent up to a label renaming. Therefore properties derived from the critical graph of $\bar{A}$ are also valid for A.

The critical graph of $\bar{A}$ is given in Figure 5.8 for the three different possibilities of the eigenvalue: $\lambda = \tau_\gamma = \tau_\delta^{\otimes m}$, $\lambda = \tau_\delta^{\otimes m} > \tau_\gamma$, and $\lambda = \tau_\gamma > \tau_\delta^{\otimes m}$.

Consider the following assumption:

Assumption 2 (A2) $\tau_\gamma \leq \tau_\delta^{\otimes m}$.

Lemma 5.6

If assumption A2 is verified then the critical graph of $\mathcal{G}^{\mathrm{c}}(A)$ (and $\mathcal{G}^{\mathrm{c}}(\bar{A})$) has one strongly connected subgraph.

Proof: We consider three cases:

- Case 1: $\tau_\gamma = \tau_\delta^{\otimes m} = \lambda$
- Case 2: $\tau_\gamma < \tau_\delta^{\otimes m} = \lambda$
- Case 3: $\tau_\delta^{\otimes m} < \tau_\gamma = \lambda$

For each of these cases the critical graph is derived from the precedence graph and it is shown that for the first two cases (for which Assumption A2 holds) the critical graph has a single strongly connected subgraph.

- Case 1: $\tau_\gamma = \tau_\delta^{\otimes m} = \lambda$
  In this situation the circuits presented in Figures 5.6-a1 and 5.6-a2 all belong to the critical graph since their weights are τ_γ or $\tau_\delta^{\otimes m}$ and thus both equal to the max-plus eigenvalue λ. Note that any circuit of the type c_1 of length l made from the nodes of t_{ℓ_m}, illustrated in Figure 5.6-a1, has an average weight of

$$\frac{|c_1|_w}{|c_1|_1} = \frac{\left(\tau_\delta^{\otimes m}\right)^{\otimes l}}{l} = \tau_\delta^{\otimes m} = \lambda, \tag{5.55}$$

  and as such also belongs to the critical graph. Any other circuit in the precedence graph of $\bar{A}$ must pass through at least one node of t_{ℓ_m}, as illustrated in Figures 5.7-b, 5.7-c1, and 5.7-c2 (with the exception of the self-loops in Figure 5.6-a3 and the circuits in Figure 5.6-a4 that we do not consider since their weights are e and $\tau_\gamma/2$, i.e., both less then λ). Additionally, arcs starting in nodes from a group t_{ℓ_q}

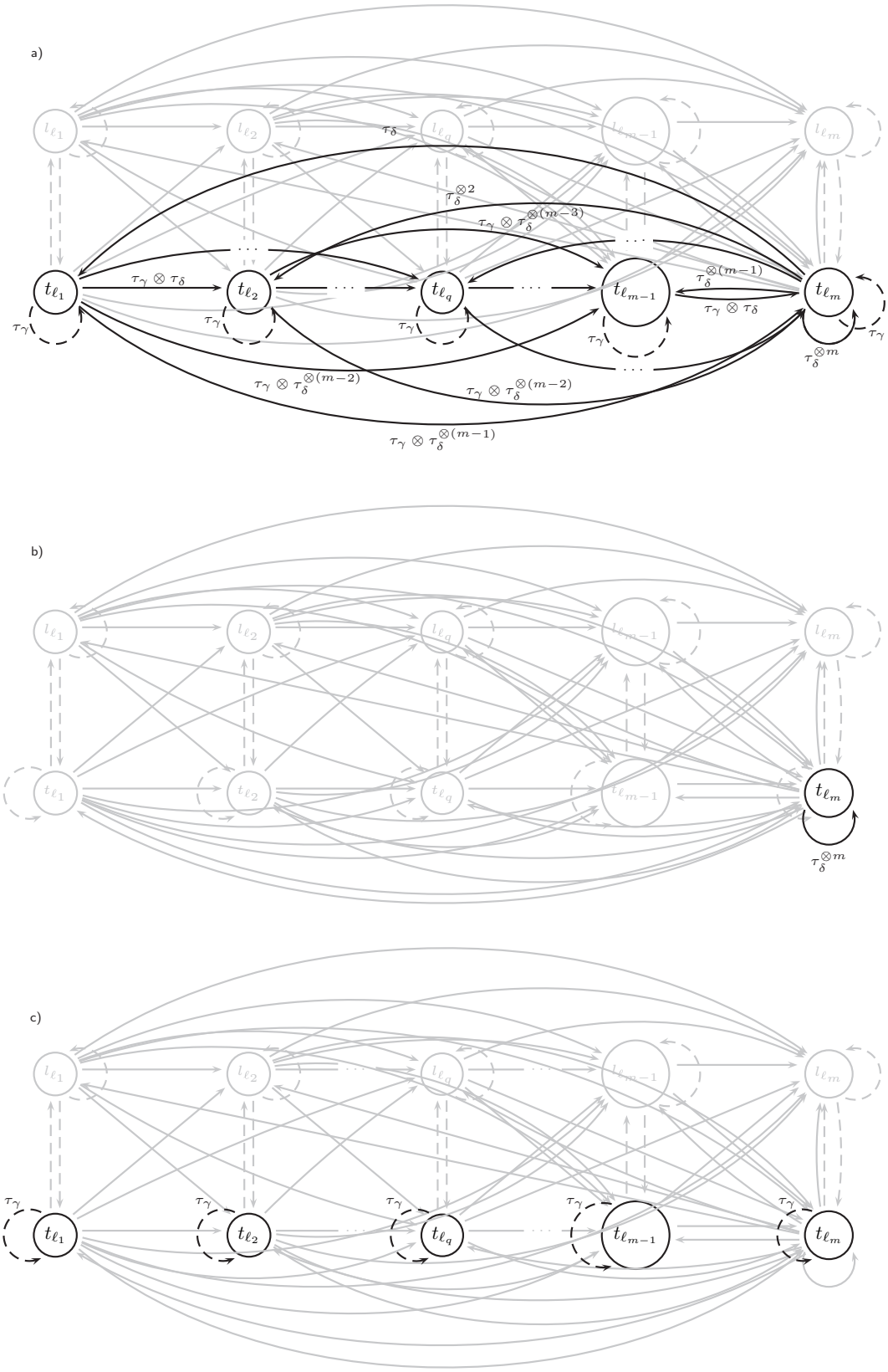

Figure 5.8: Critical graphs of the system matrix $\bar{A}$. a) Case 1: $\tau_\gamma = \tau_\delta^{\otimes m} = \lambda$. b) Case 2: $\tau_\gamma < \tau_\delta^{\otimes m} = \lambda$. c) Case 3: $\tau_\delta^{\otimes m} < \tau_\gamma = \lambda$.

with $q < m$ are only connected to nodes in $t_{\ell_{q+p}}$ for $p \geq 0$ (or $l_{\ell_{q+p}}$). This is again illustrated in Figures 5.7-a, 5.7-c1, and 5.7-c2. Let $t_{[\ell_q]_i}$ denote the i-th element of t_{ℓ_q}. Consider the circuit of type

$$c_2 : t_{[\ell_m]_i} \to t_{[\ell_q]_j} \to t_{[\ell_m]_i}, \tag{5.56}$$

with $q < m$. The average weight is (with $\tau_\gamma = \tau_\delta^{\otimes m}$)

$$\frac{|c_2|_w}{|c_2|_1} = \frac{\tau_\delta^{\otimes q} \otimes \tau_\gamma \otimes \tau_\delta^{\otimes(m-q)}}{2} = \frac{\tau_\delta^{\otimes m} \otimes \tau_\gamma}{2} = \lambda. \tag{5.57}$$

Circuit c_2 is thus also in the critical graph. For the general circuit of the type

$$c_3 : t_{[\ell_m]_i} \to \underbrace{t_{[\ell_{q_1}]_{j_1}} \to t_{[\ell_{q_2}]_{j_2}} \to \cdots \to t_{[\ell_{q_l}]_{j_l}}}_{l \text{ nodes}} \to t_{[\ell_m]_i}, \tag{5.58}$$

with $q_1 < q_2 < \cdots < q_l < m$, the average weight is

$$\frac{|c_3|_w}{|c_3|_1} = \frac{\tau_\gamma^{\otimes l} \otimes \tau_\delta^{\otimes q_1} \otimes \tau_\delta^{\otimes(q_2-q_1)} \otimes \cdots \otimes \tau_\delta^{\otimes(m-q_l)}}{l+1} \tag{5.59}$$

$$= \frac{\tau_\gamma^{\otimes l} \otimes \tau_\delta^{\otimes m}}{l+1} = \lambda. \tag{5.60}$$

Hence, circuits of type c_3 are also part of the critical graph.

However, any circuit that passes through any node in l_{ℓ_q}, for any q, will never be in the critical graph. This is due to the fact that arcs within touchdown nodes of different leg groups yield a higher weight:

$$t_{[\ell_q]_i} \to t_{[\ell_p]_j} \qquad \text{weight: } \tau_\gamma \otimes \tau_\Delta^{\otimes(q-p)} \tag{5.61}$$

$$t_{[\ell_q]_i} \to l_{[\ell_p]_j} \qquad \text{weight: } \tau_\mathrm{g} \otimes \tau_\Delta^{\otimes(q-p)} \tag{5.62}$$

$$l_{[\ell_q]_i} \to t_{[\ell_p]_j} \qquad \text{weight: } \tau_\mathrm{f} \otimes \tau_\Delta^{\otimes(q-p)} \tag{5.63}$$

$$l_{[\ell_q]_i} \to l_{[\ell_p]_j} \qquad \text{weight: } \tau_\Delta^{\otimes(q-p)}. \tag{5.64}$$

As such, a path that connects a touchdown node to a lift off node "loses" $\tau_\gamma - \tau_\mathrm{g} = \tau_\mathrm{f}$ from the maximum possible weight, a path from lift off to lift off nodes loses τ_γ, and a path from lift off nodes to touchdown nodes loses τ_g in weight. This can also be observed in the structure of $\bar{A}$, in equation (5.39), where the sub matrix $\tau_\mathrm{f} \otimes (\tau_\mathrm{g} \otimes W \oplus V)$ overcomes the sub matrices $\tau_\mathrm{g} \otimes W \oplus V$, $\tau_\mathrm{f} \otimes W$, and W. Consider the circuit of type c_4:

$$c_4 : t_{[\ell_m]_i} \to t_{[\ell_p]_{j_0}} \to l_{[\ell_{p+q}]_{j_q}} \to t_{[\ell_m]_i}, \tag{5.65}$$

then

$$\frac{|c_4|_w}{|c_4|_1} = \frac{\tau_\delta^{\otimes p} \otimes (\tau_\mathrm{g} \otimes \tau_\delta^{\otimes q}) \otimes (\tau_\mathrm{f} \otimes \tau_\delta^{\otimes(m-(p+q))})}{3} \tag{5.66}$$

$$= \frac{\tau_\gamma \otimes \tau_\delta^{\otimes m}}{3} < \lambda. \tag{5.67}$$

Since all the nodes in the critical graph are connected (they are all touchdown nodes) we conclude that for the case $\tau_\gamma = \tau_\delta^{\otimes m} = \lambda$ the critical graph of $\bar{A}$ has a single strongly connected subgraph. Figure 5.8-a illustrates the complete critical graph of $\bar{A}$ for this case.

- Case 2: $\tau_\gamma < \tau_\delta^{\otimes m} = \lambda$
  In this situation only circuits of the type c_1 are part of the critical graph. Circuits of the type c_2 or c_3 are not part of the critical graph. Figure 5.8-b illustrates the resulting critical graph of $\bar{A}$. Since all the nodes of t_{ℓ_m} are connected to each other we conclude that for the case $\tau_\gamma < \tau_\delta^{\otimes m} = \lambda$ the critical graph of $\bar{A}$ has a single strongly connected subgraph.

- Case 3: $\tau_\delta^{\otimes m} < \tau_\gamma = \lambda$
  In this situation the critical graph of $\bar{A}$ does not have a single strongly connected subgraph. Figure 5.8-c illustrates this situation, which we document here without proof.

□

This lemma leads to the following theorem about the uniqueness of the eigenvalue and eigenvector:

Theorem 5.7 *Given assumptions A1 and A2, the max-plus eigenvalue λ of the system matrix A (and $\bar{A}$), defined by equation (5.50), is unique, and the max-plus eigenvector v of A (and $\bar{A}$), defined by equations (5.51)–(5.52) is unique up to a max-plus scaling factor.*

Proof: According to Lemma 5.4 the matrix A is irreducible, and as such it has a unique max-plus eigenvalue. According to Lemma 5.6 the critical graph of $\mathcal{G}^{\mathrm{c}}(A)$ has a single strongly connected subgraph, and as such its max-plus eigenvector is unique up to a max-plus scaling factor (see Theorem 3.101 in the work of Baccelli et al. [3]). □

5.3.3 Coupling time

Theorem 2.3 on page 17 describes an important property of max-plus-linear systems when the system matrix A is irreducible: it guarantees the existence of an autonomous steady-state regime that is achieved in a number of finite steps k_0, called the *coupling time*. Computing the coupling time is very important for the application of legged locomotion since it provides the number of steps a robot needs to take to reach steady state after a gait transition or a perturbation.

Lemma 5.8
Given assumptions A1, A2, the coupling time for the max-plus-linear system defined by equation (5.23) is $k_0 = 2$ with cyclicity $c = 1$.

Proof: See Lopes et al. [68]. □

Next gait switching is discussed.

5.4 Gait switching

We now address the problem of choosing gaits and their transition parameters when changing rhythms. In Section 5.4.1 we discuss how to choose gaits to obtain the best possible transitions given the models presented earlier. In Section 5.4.2 we introduce a new scheme that results in an equal stance time for all legs during transitions. This result is used in Section 5.4.3 to enable constant acceleration/deceleration in legged robots while switching gaits.

5.4.1 Compatible gaits for switching

Let G_n, called the *gait space*, be the set of all gaits defined according to expression (5.14) for an n-legged robot. Also, let $\mathcal{A}_n$ be the set of all system matrices for gaits generated from (5.14) with equation (5.36):

$$\mathcal{A}_n = \{A(1), \ldots, A(n)\}, \tag{5.68}$$

One can write the switching max-plus linear system

$$x(k+1) = A^{\vartheta(k)} \otimes x(k), \tag{5.69}$$

where $\vartheta(k)$ is a switching function whose value is determined by the supervisory controller based on the desired gait. By construction, gait switching is kinematic stance stable, in the sense that for two different gaits that swing at most q_i and q_j legs simultaneously, we will have at most $\max(q_i, q_j)$ legs swinging during the transition between both. For example during the transition between a walk and a trot on a quadruped robot, no more than two legs can swing simultaneously (note that since we are not taking into consideration the dynamics of the robot this measure of "stability" applies only to the discrete-event supervisory controller). By looking at the definition of a gait in expression (5.14) it is clear that the size of the gait space G_n is combinatorial in n (in fact $\#\mathsf{G}_n = n! \times (2^{(n-1)} - 1)$, i.e. the number of permutations of n elements times the number of possible set partitions, excluding the partition consisting of a set with n elements). However, different representations for a gait as an ordered set of ordered sets can lead to the same exact robot physical motion behavior, as in the following example:

$$\mathrm{G}_1 = \{1,2\} \prec \{3,4\} \prec \{5,6\} \tag{5.70}$$
$$\mathrm{G}_2 = \{2,1\} \prec \{3,4\} \prec \{5,6\} \tag{5.71}$$
$$\mathrm{G}_3 = \{5,6\} \prec \{1,2\} \prec \{3,4\} \tag{5.72}$$
$$\mathrm{G}_4 = \{4,3\} \prec \{6,5\} \prec \{2,1\} \tag{5.73}$$
$$\ldots \tag{5.74}$$

The difference lies in the fact that the transition between the above defined gaits and a new different gait, say $\mathrm{G}_5 = \{3,4,6\} \prec \{1,2,5\}$, will result in a different transient behavior, as illustrated in the examples of Figures 5.9.a) and 5.9.b). This poses the question of how to optimally switch gaits, in the sense of minimizing the variation of the leg stance time

during gait switching. For applications of climbing robots [42] it is fundamental that all legs exert the same force on the attaching wall at all times, thus motivating constant foot velocity (viewed from a frame attached to the robot). The same is valid for walking robots, as different leg velocities can result in turning moments that can make the legged platform unstable. For the n-legged robot with gaits represented by (5.14) suppose the gait switching mechanism consists in moving a single leg from one group of legs ℓ_i to a different group of legs ℓ_j with $i,j \in \{1,\ldots,\}$. By inspecting the max-plus eigenvector (thus assuming steady-state behavior), one can observe that the moment that a leg in the set ℓ_i lifts off the ground happens at the time instant

$$(\tau_{\mathrm{f}} \otimes \tau_{\Delta})^{\otimes i}, \tag{5.75}$$

assuming the cycle starts at zero time. Analogously, for a leg in the set ℓ_j we get the lift-off time to be:

$$(\tau_{\mathrm{f}} \otimes \tau_{\Delta})^{\otimes j}. \tag{5.76}$$

Moving a leg from the set ℓ_i to the set ℓ_j results in a change of lift-off time of

$$(\tau_{\mathrm{f}} \otimes \tau_{\Delta})^{\otimes (j-i)}. \tag{5.77}$$

If $j > i$, then the switching leg will stay in stance for an extra $(\tau_{\mathrm{f}} \otimes \tau_{\Delta})^{\otimes (j-i)}$ time units during the transition to synchronize with the new leg group. This is always the case since the time of flight τ_{f} is fixed. If $j < i$ then all the legs in the original group of the switching leg will have their lift-off times postponed by $(\tau_{\mathrm{f}} \otimes \tau_{\Delta})^{\otimes (i-j)}$ time units. Thus, the larger the magnitude of $j-i$ the larger the stance time variation during the transition will be. For instance, the gait transition of

$$\{1,\mathbf{2}\} \prec \{3,4\} \prec \{5\} \prec \{6\} \rightarrow \{1\} \prec \{\mathbf{2},3,4\} \prec \{5\} \prec \{6\} \tag{5.78}$$

has less stance time variation than the transition

$$\{1,\mathbf{2}\} \prec \{3,4\} \prec \{5\} \prec \{6\} \rightarrow \{1\} \prec \{3,4\} \prec \{\mathbf{2},5\} \prec \{6\}. \tag{5.79}$$

The difference is in the position of leg 2, indicated in bold, after the transition. The same is true when changing the number of leg groups, e.g. the gait transition of

$$\{1,2,3\} \prec \{4,5,6\} \rightarrow \{1,2\} \prec \{3,4\} \prec \{5,6\} \tag{5.80}$$

has less stance time variation than the transition

$$\{1,2,3\} \prec \{4,5,6\} \rightarrow \{5,6\} \prec \{1,2\} \prec \{3,4\}. \tag{5.81}$$

This provides a simple mechanism for choosing gaits without requiring to search the gait space for all structurally equivalent gaits. Figure 5.9 illustrates the comparison of a non-optimal gait switch (in Figure 5.9.a) with an optimal one (in Figure 5.9.b). To quantify the quality of a gait transition, we introduce the following measure:

$$\bar{\sigma} = \frac{1}{\tau_{\mathrm{g}}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\bar{\tau}_{\mathrm{g}i} - \bar{\bar{\tau}}_{\mathrm{g}})^2}, \tag{5.82}$$

where $\bar{\tau}_{gi}$ is the true stance time of leg i, and $\bar{\bar{\tau}}_g$ is the average stance time for all legs, both during the transition. In formula (5.82) we divide the unbiased standard deviation of $\bar{\tau}_{gi}$ by the desired stance time τ_g to obtain a non-dimensional measure. If $\bar{\sigma} = 0$ then the transition maintains a constant stance time for all legs. Note that minimizing $\bar{\sigma}$ results in minimizing the variation of the foot velocities during stance (assuming a constant foot velocity for the stance phase range).

5.4.2 Variable swing time, constant stance model

As shown before, by selecting the leg indices in the proper way when switching a gait, one can achieve a better switching behavior. However, by construction, since the synchronization happens at the lift-off time, during gait transitions some legs will inevitably stay longer on the ground, which can cause undesired behavior of the robotic platform, because the legs will move at different velocities causing some legs to slip, or the robot to turn slightly. We now show that by manipulating the flight time of each leg independently one can achieve a unique stance time for all legs under well-defined assumptions. Consider the new model:

$$\begin{bmatrix} t(k) \\ l(k) \end{bmatrix} = \left[\begin{array}{c|c} \mathcal{E} & R \\ \hline P & \mathcal{E} \end{array}\right] \otimes \begin{bmatrix} t(k) \\ l(k) \end{bmatrix} \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline \tau_g \otimes E \oplus Q & E \end{array}\right] \otimes \begin{bmatrix} t(k-1) \\ l(k-1) \end{bmatrix}, \tag{5.83}$$

where the diagonal matrix R represents different swing times:

$$R = \begin{bmatrix} \tau_{f1} & \varepsilon & \cdots & \varepsilon \\ \varepsilon & \tau_{f2} & & \\ \vdots & & \ddots & \\ \varepsilon & & & \tau_{fn} \end{bmatrix}. \tag{5.84}$$

Following the definition (5.35) let

$$\bar{A}(\mathrm{G}, R, \tau_g, \tau_\Delta) := \left[\begin{array}{c|c} \mathcal{E} & R \\ \hline P_{\mathrm{G},\tau_\Delta} & \mathcal{E} \end{array}\right]^* \otimes \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline \tau_g \otimes E \oplus Q_{\mathrm{G},\tau_\Delta} & E \end{array}\right], \tag{5.85}$$

where the matrices $P_{\mathrm{G},\tau_\Delta}$ and $Q_{\mathrm{G},\tau_\Delta}$ are constructed according to expressions (5.20) and (5.21), respectively, for a gait G. Then, the system matrix of (5.36) is parameterized as:

$$\bar{A}(\mathrm{G}, \tau_f \otimes E, \tau_g, \tau_\Delta), \tag{5.86}$$

and the resulting system matrix of (5.83) is parameterized by:

$$\bar{A}(\mathrm{G}, R, \tau_g, \tau_\Delta). \tag{5.87}$$

Let $\max_v : \mathbb{R}^n \to \mathbb{R}$ and $\min_v : \mathbb{R}^n \to \mathbb{R}$ be operators on vectors that return the maximum or the minimum element of a vector, respectively. Now consider two different gaits G_1 and G_2 with respective eigenvectors $v_{\mathrm{G}_1} = [t_{\mathrm{G}_1}^\top \; l_{\mathrm{G}_1}^\top]^\top$ and $v_{\mathrm{G}_2} = [t_{\mathrm{G}_2}^\top \; l_{\mathrm{G}_2}^\top]^\top$. During a transition from gait G_1 to the gait G_2 the extra time each leg will stay in stance can be computed by:

$$\pi = (l_{\mathrm{G}_2} - t_{\mathrm{G}_1}) - \min_v (l_{\mathrm{G}_2} - t_{\mathrm{G}_1}). \tag{5.88}$$

A transition system matrix $\bar{A}(\mathrm{G}_1, R_1, \tau_\mathrm{g}, \tau_\Delta)$ can be constructed such that for each leg an element of the "extra time" vector $\pi \in \mathbb{R}^n_\varepsilon$ is subtracted from the flight time τ_f so that in the next cycle, now using gait G_2, will make the real stance time $\bar{\tau_\mathrm{g}}$ the same for each leg. Note that this is only possible if

$$\tau_{\mathrm{fG}_1} \geq \max_v(\pi), \tag{5.89}$$

where τ_{fG_1} is the swing time parameter for gait G_1. If that is not the case, then an additional transition matrix, now using gait G_2, can be constructed as $\bar{A}(\mathrm{G}_2, R_2, \tau_\mathrm{g}, \tau_\Delta)$ such that the time that cannot be subtracted from the transition matrix R_1 is subtracted from the matrix R_2. The resulting transition algorithm is summarized as follows:

1. Given two gaits G_1 and G_2 compute π via (5.88).

2. If $\tau_{\mathrm{fG}_1} \geq \max_v(\pi)$ then compute the vector:

$$\pi_{\mathrm{t}1} = [(\tau_{\mathrm{fG}_1} - [\pi]_1) \ \cdots \ (\tau_{\mathrm{fG}_1} - [\pi]_n)]^\top, \tag{5.90}$$

   and the system matrix

$$\bar{A}(\mathrm{G}_1, \mathrm{diag}(\pi_{\mathrm{t}1}), \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}). \tag{5.91}$$

   where $\mathrm{diag} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ returns a matrix with the elements of a vector on the leading diagonal. The transition sequence is obtained by the following sequence of system matrices:

$$\begin{aligned}
A(k-p) &= \bar{A}(\mathrm{G}_1, \tau_{\mathrm{fG}_1} \otimes E, \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.92)\\
&\vdots && (5.93)\\
A(k-1) &= \bar{A}(\mathrm{G}_1, \tau_{\mathrm{fG}_1} \otimes E, \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.94)\\
A(k) &= \bar{A}(\mathrm{G}_1, \mathrm{diag}(\pi_{\mathrm{t}1}), \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.95)\\
A(k+1) &= \bar{A}(\mathrm{G}_2, \tau_{\mathrm{fG}_2} \otimes E, \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}) && (5.96)\\
&\vdots && (5.97)\\
A(k+p) &= \bar{A}(\mathrm{G}_2, \tau_{\mathrm{fG}_2} \otimes E, \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}). && (5.98)
\end{aligned}$$

3. If $\tau_{\mathrm{fG}_1} < \max_v(\pi)$ then create two transition matrices

$$\bar{A}(\mathrm{G}_1, \mathrm{diag}(\pi_{\mathrm{t}1}), \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}), \tag{5.99}$$

   and

$$\bar{A}(\mathrm{G}_2, \mathrm{diag}(\pi_{\mathrm{t}2}), \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}), \tag{5.100}$$

   where

$$[\pi_{\mathrm{t}1}]_i = \max(\min([\pi]_i, \tau_{\mathrm{fG}_1}), \tau_{\mathrm{f,min}}), \tag{5.101}$$

with $\tau_{\mathrm{f,min}} > 0$ the minimum leg swing time, and

$$[\pi_{\mathrm{t}2}]_i = \tau_{\mathrm{fG}_2} - ([\pi_{\mathrm{t}1}]_i - [\pi]_i) - \min_v(\pi_{\mathrm{t}1} - \pi). \tag{5.102}$$

The transition sequence is obtained by the following sequence of system matrices:

$$\begin{aligned}
A(k-p) &= \bar{A}(\mathrm{G}_1, \tau_{\mathrm{fG}_1} \otimes E, \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.103)\\
&\vdots && (5.104)\\
A(k-1) &= \bar{A}(\mathrm{G}_1, \tau_{\mathrm{fG}_1} \otimes E, \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.105)\\
A(k) &= \bar{A}(\mathrm{G}_1, \mathrm{diag}(\pi_{\mathrm{t}1}), \tau_{\mathrm{gG}_1}, \tau_{\Delta\mathrm{G}_1}) && (5.106)\\
A(k+1) &= \bar{A}(\mathrm{G}_2, \mathrm{diag}(\pi_{\mathrm{t}2}), \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}) && (5.107)\\
A(k+2) &= \bar{A}(\mathrm{G}_2, \tau_{\mathrm{fG}_2} \otimes E, \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}) && (5.108)\\
&\vdots && (5.109)\\
A(k+p) &= \bar{A}(\mathrm{G}_2, \tau_{\mathrm{fG}_2} \otimes E, \tau_{\mathrm{gG}_2}, \tau_{\Delta\mathrm{G}_2}). && (5.110)
\end{aligned}$$

5.4.3 Variable velocity

Variable velocity can be achieved by scaling the time τ. As presented earlier, the actuator reference trajectories q_{ref} are generated by the following equation:

$$q_{\mathrm{ref}}(\tau) = g(p, \theta_{\mathrm{ref}}(\tau, S(\tau))). \tag{5.111}$$

By introducing a "time modulating" function $\alpha : \mathbb{R} \to \mathbb{R}$ we obtain a new reference phase generator:

$$q_{\mathrm{ref}}(\tau) = g(p, \theta_{\mathrm{ref}}(\alpha(\tau), S(\alpha(\tau)))). \tag{5.112}$$

A constant accelerating robot can be obtained by choosing $\alpha(\tau) = a\tau$ where a is the desired acceleration. Taking into account the minimum time required for a leg to swing, gait switching can be triggered automatically when due to the acceleration the minimum flight time is reached (it is assumed that the gaits are specified such that Assumptions A1 and A2 are satisfied, therefore gaits with less leg groups have a longer flight time for a given ground time compared to gaits with more leg groups, so switching to a gait with less leg groups when the minimum flight time is reached allows the robot to continue accelerating).

5.5 Simulations

The robots Zebro and RQuad, which are morphologically identical to RHex [80], are utilized for experimental validation and are illustrated in Figure 5.3. The physical robots have a single motor per leg, and as such the dimensions of the vectors q_{ref} and θ match.

5.5.1 Simulation of the max-plus gait scheduler

In this subsection simulations of the max-plus gait scheduler when performing gait transitions and when accelerating are presented.

Figure 5.9.a) illustrates non-optimal gait switches for the hexapod robot Zebro. The order of gaits is:

$$\{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\} \rightarrow \{5,2\} \prec \{3,6\} \prec \{1,4\} \rightarrow \\ \{2,3,6\} \prec \{1,4,5\} \rightarrow \{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\}.$$

The non-dimensional standard deviation $\bar{\sigma}$ for transition 1 is $(\bar{\sigma})_1 = 0.57$, for transition 2 is $(\bar{\sigma})_2 = 0.45$, and for transition 3 is $(\bar{\sigma})_3 = 0.80$.

Figure 5.9.b) illustrates optimal gait switches with fixed flight time τ_f for the hexapod robot Zebro. The order of gaits is:

$$\{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\} \rightarrow \{1,4\} \prec \{5,2\} \prec \{3,6\} \rightarrow \\ \{1,4,5\} \prec \{2,3,6\} \rightarrow \{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\}.$$

The non-dimensional standard deviation for transition 1 is $(\bar{\sigma})_1 = 0.14$, for transition 2 is $(\bar{\sigma})_2 = 0.33$, and for transition 3 is $(\bar{\sigma})_3 = 0.19$. These deviations are clearly much lower than those of the non-optimal gait switches.

Figure 5.9.c) illustrates an example transition with constant stance times τ_g and different flight times τ_f for each leg during the transitions, highlighted by the green shades of color. The order of gaits is the same as for Figure 5.9.b):

$$\{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\} \rightarrow \{1,4\} \prec \{5,2\} \prec \{3,6\} \rightarrow \\ \{1,4,5\} \prec \{2,3,6\} \rightarrow \{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\}.$$

In this case by adjusting the flight times the non-dimensional standard deviation is $(\bar{\sigma})_1 = (\bar{\sigma})_2 = (\bar{\sigma})_3 = 0$.

Figure 5.9.d) illustrates a hexapod robot that is constantly accelerating and performing gait transitions with variable flight times τ_f. In this case there are only two gait transitions:

$$\{1\} \prec \{4\} \prec \{5\} \prec \{2\} \prec \{3\} \prec \{6\} \rightarrow \{1,4\} \prec \{5,2\} \prec \{3,6\} \rightarrow \{1,4,5\} \prec \{2,3,6\}.$$

5.6 Summary

In this chapter an overview was given on current approaches for the modeling of locomotion patterns for legged locomotion namely central pattern generators and Buehler clocks. The main drawbacks of these approaches are the limited knowledge on the transitional behavior during gait switches and ensuring stable gait switches. Therefore a generalization of the Buehler clock approach was proposed, where the each foot's interaction with the ground is modeled via switching max-plus-linear systems. The advantage of the switching max-plus-linear description is that the techniques of max-plus algebra can be used to analyze the behavior of the gaits and the transitional behavior during gait switches.

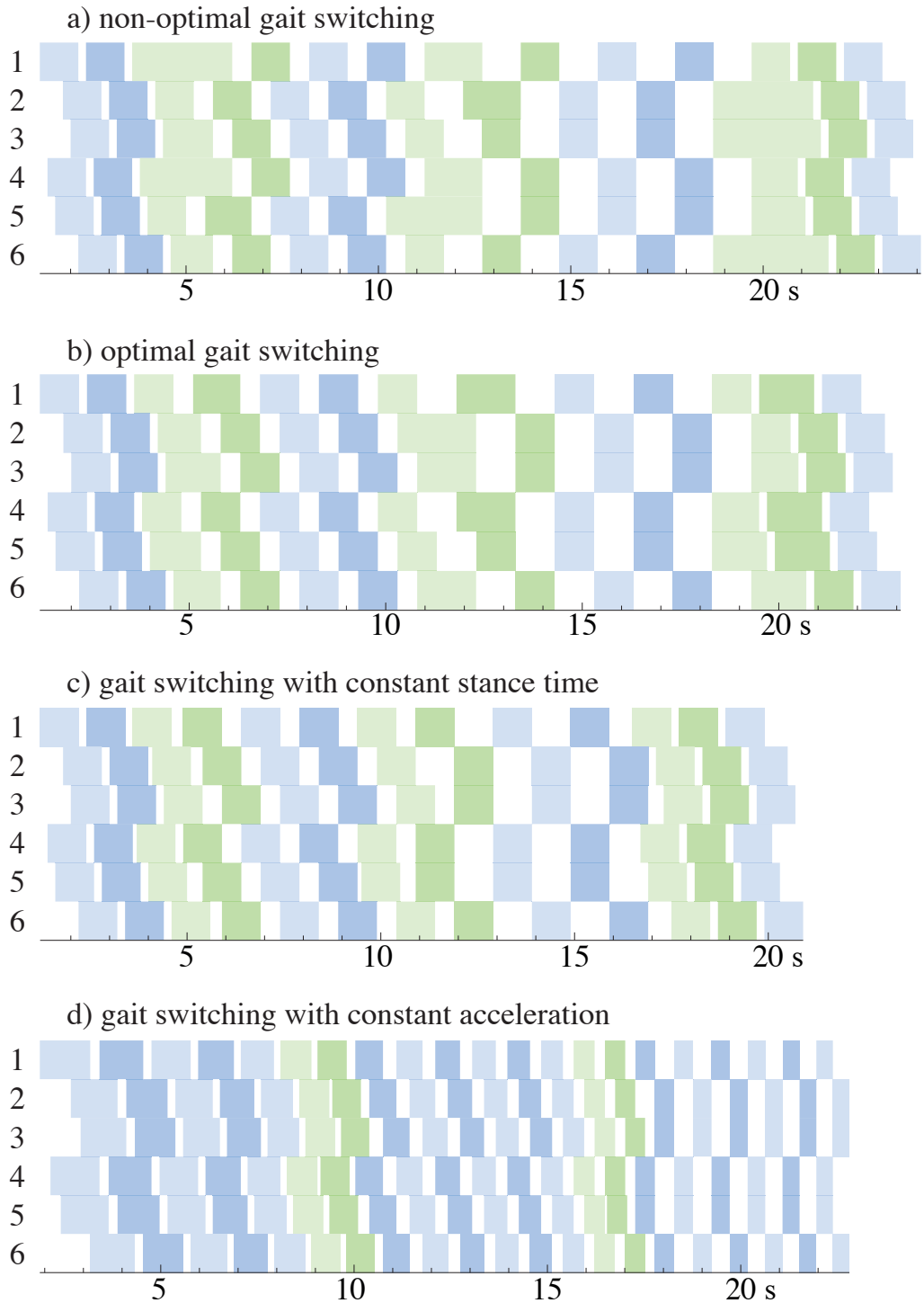

Figure 5.9: Various real walking experiments executed using the Zebro robot. The color bars represent legs in stance and the transparent areas represent leg swing. Transition steps are indicated by the green shades of color outlines. Sub-figure a) non-optimal gait switches. Sub-figure b) optimal gait transitions with fixed τ_f. Sub-figure c) optimal gait switch with transitions with variable τ_f. Sub-figure d) gait transitions with constant acceleration.

For the switching max-plus linear systems describing the gait reference generator important structural properties of the max-plus system matrix are obtained in closed-form. On the graph side, the node-reduction procedure has allowed for a compact graphical representation of the system matrix describing the gait for an arbitrary number of legs and leg groups. This was achieved by grouping all nodes per leg group into a touchdown and liftoff node and by making use of repeating patterns in the graph. We have shown that the eigenvector is unique (up to a scaling factor) if the design parameters for the gaits are chosen such that two assumptions are met. The uniqueness of the eigenvector was proven by showing that the critical graph of the system matrix consists of one strongly connected subgraph.

With the steady state behavior uniquely defined for all gaits we were able to determine optimal gait switches. Furthermore a method was developed that manipulated the flight times of the legs during gait switches to ensure the ground time of all legs is the same and to speed up the gait switch.

Finally a method was developed to allow the robot to accelerate (or decelerate) at a constant pace by adjusting the timing function of the robot.

In the future we will research instant gait transitions (without waiting for a cycle to finish), and the modeling of more general gaits, including those with an aerial phase. Another topic of future research should be the automatic switching of gaits and adjustment of the gait parameters based on the current environmental circumstances.

Chapter 6

Conclusions and recommendations

6.1 Conclusions

The main contributions in this thesis can be summarized as follows:

- A switching max-plus-linear model of the railway traffic that can be used for network-wide on-line railway traffic management has been presented in this thesis. The modeling of the railway traffic was extended compared to models in the literature by extending the control actions to allow the trains to switch tracks when there are multiple tracks between stations. Furthermore a new max-plus matrix formulation is introduced. Finally the modeling of the railway traffic is extended with extra constraints for freight trains.

- With the use of max-plus algebraic theories and the newly introduced matrix formulation the implicit switching max-plus linear model of the railway traffic network has been converted into its explicit form. A reduction method was developed to reduce the size and complexity of the explicit model, resulting in a decrease in the computation time needed to solve the rescheduling problem using the explicit model compared to the implicit model. For a model consisting of the largest part of the Dutch railway network the average computation time of a single step of the model predictive control problem was reduced by a factor 5.44 for a one-hour prediction horizon when trying to minimize the sum of delays and by a factor 10 when trying to minimize the sum of arrival delays. For a two-hour prediction horizon the average computation time did not decrease for the sum of delays, but the maximum computation time was 6.15 times lower. For the sum of arrival delays the average computation was reduced by a factor 2.55. The proposed conversion to the explicit model and subsequent reduction of the model clearly reduces the time needed to solve the dispatching problem using that model, and shows the advantage of the explicit model when trying to minimize a subset of the delays such as only the arrival delays. For larger models and larger prediction horizons the improvement in the computation time will likely be even lower. The conversion to the explicit model and the subsequent reduction is therefore more appropriate for smaller models.

- Several distributed model predictive (DMPC) control methods have been developed in this thesis and extensively tested in various case studies using a model of the complete Dutch railway network for various lengths of the prediction horizon.

  In the first case study a model of the Dutch railway network containing all Nederlandse Spoorwegen trains as described in the timetable of 2011 was used. The network was partitioned into four parts for the DMPC approaches. A prediction horizon of 60 minutes was used and the approaches were compared in terms of computation time and delay reduction for a single time-instant for 1000 scenarios. The result was that DMPC method 4 was the best by finding solutions that on average only had 1.27% more delays than the centralized MPC approach, while the computation time was much lower than for the centralized approach and among the lowest for the DMPC approaches.

  In the second case study the centralized problem was partitioned into 2, 3, 4, 6, and 8 parts for the DMPC approaches and tested for prediction horizons of 30, 45, 60, and 75 minutes. Instead of comparing the performance of a single time instant for multiple scenarios, now the performance was compared for a three hour window, where the controller was in a closed loop, such that the control actions each time instant are implemented and the consequences of those control actions affect the current and future time instants. Therefore, each minute a new optimization had to be solved and implemented. The total delay in this three hour was compared for all four approaches, for all five partitions, and four prediction horizon lengths, for 100 scenarios. In the end the DMPC methods 1 and 4 performed comparable in both computation time and delay reduction and when considering limits on the maximum computation time outperformed the centralized MPC approach. The benefit of increasing the prediction horizon on the delay reduction lessens as the prediction horizon becomes larger, while the computation time grows exponentially. Based on the case studies we performed it seems that a prediction horizon with length between 60 and 75 minutes gives the best delay reduction compared to the computation time. There are several possible explanations for the lower increase in the delay reduction for increasing prediction horizons. First of all the predictions further ahead in the future are less accurate and more likely to change when new information becomes available in the future. A second explanation could be that due to the buffer times in dwell and running times, the delays are absorbed and for longer prediction horizons more delays will have been absorbed at the end of the horizon and for smaller delays the controller is less likely to take control actions.

- In this thesis the steady state cyclic behavior of the max-plus-linear systems describing the gaits, and the transition to the steady state cyclic behavior, are analyzed. We have proven that, given mild conditions, the eigenvector is unique (up to a scaling factor). This was done by showing that the critical graph of the system matrix consists of one strongly connected subgraph. In order to show this we had to simplify the critical graph. To do that we proposed a node-reduction procedure that has allowed for a compact graphical representation of the system matrix describing the gait for an arbitrary number of legs and leg groups. This was achieved by

grouping all nodes per leg group into a touchdown and liftoff node and by making use of repeating patterns in the graph. The graph of the system matrix was used to determine that, if the design parameters for the gaits are chosen such that two assumptions are met, the critical graph consists of one strongly connected subgraph which is equivalent to proving the uniqueness of the max-plus eigenvector.

- With the steady state behavior uniquely defined for all gaits we were able to determine gait switches that minimized the variation in ground time between the legs. Furthermore a method was developed that manipulated the flight times of the legs during gait switches to ensure the ground time of all legs is the same and to speed up the gait switch.

- Finally a method was developed to allow the robot to accelerate (or decelerate) at a constant pace by adjusting the timing function of the robot.

6.2 Recommendations

In this section recommendations for future research are presented. First we will discuss some of the open problems in railway traffic management, after that we will propose some research directions for legged locomotion and finally we will give some general recommendations for future research.

6.2.1 Railway traffic management

In this thesis we presented a framework for on-line railway traffic management using a switching max-plus-linear model of the railway traffic and (distributed) model predictive control to determine a new schedule for the railway traffic in the case of delays. As a cost function for the criterion the sum of (arrival) delays was used. This cost function is based on the event times of the trains and does not consider the passengers in the trains. Furthermore we assume there are no major disruptions in the network such as blocked tracks, or broken-down trains. Railway traffic management includes many more facets than just the schedule of departures and arrivals of the trains. Other facets are the personnel schedules and the rolling stock schedule. To find a complete solution to the problem of railway traffic management all of these open problems need to be tackled.

- Explicit model and reduction method
  Currently the conversion of the implicit model to the explicit model, and the application of the reduction method, is done off-line and for larger models can take several minutes to complete. Furthermore the reduction method is based on a limitation of the control actions such that the solutions found are only optimal if the maximum deviation from the nominal timetable is limited. But if the conversion from the implicit to the explicit model, and the reduction method, could be sped up such that it can be applied on-line the reduction method can be changed such that the limitation of the control actions is based on a maximum deviation from the last

known solution. By making this change the solutions found with the model predictive controller using the reduced explicit model are optimal for larger deviations from the nominal timetable without increasing the problem complexity, as long as the change in situation in the network between time instants is limited.

- Explicit model and DMPC
  From the case study it is clear that the improvement in computation time from using the reduced explicit model for the model predictive controller is especially effective for smaller problems. Since we want to use global control on large networks with prediction horizons around 60 minutes the problem size is relatively large and therefore using the reduced explicit model does not yield much better computation times. With the DMPC methods we proposed the network is split up into smaller sub-networks. A future direction for research would be to combine the DMPC methods with the conversion and reduction from the implicit to the reduced explicit model.

- The effect of buffer times and delay scenarios on the delay reduction
  Based on the case studies we performed we concluded that a prediction horizon with length between 60 and 75 minutes gives the best delay reduction compared to the computation time. But the scenarios we considered all had a similar amount of delays and we only tested for one network with a fixed amount of buffer time. It is possible that for scenarios with more (and larger) delays and networks with less buffer time increasing the prediction horizon has a larger effect on the delay reduction. By testing the (D)MPC methods in case studies with scenarios with different amounts of delay and for networks with varying buffer times more general conclusions on the length of the prediction horizon can be drawn and recommendations for a large class of railway networks can be given.

- Coordination and integration of local controllers
  The coordination and integration of local controllers in the framework for railway traffic management has not been considered in this thesis, but it is an important part of the framework. Especially the feedback from the local controllers, when a solution of the global controller is not feasible, is crucial. What information must be fed back to the global controller? Are updated process times enough? Or can the local controllers propose new headway constraints, that may be needed when they reroute trains.

- Real-time monitoring
  Another part of the framework is real-time monitoring of the trains and the prediction of future process times. Research on this was done by Pavle Kecman [53], but before the control methods proposed in this thesis and the monitoring and prediction methods in his thesis can be combined the data from his monitoring and prediction methods needs to be aggregated to the macroscopic level used in our model predictive controller. Furthermore the value for the time step in the receding horizon must be chosen based on the time needed to gather and process the data,

compute the new schedule, the time needed to implement the solution, and the time needed to reiterate if the solution is found to be infeasible by the local controllers.

- Passenger delays
  Most research on railway traffic management has mainly tried to minimize train delays, but the real goal should be to minimize passenger delays. Therefore we recommend to research passenger behavior and make use of the passenger information that is currently already being gathered by railway operators using on-line check-in systems, such as the OV-chipkaart in The Netherlands, to predict the passenger flows through the network and to use this information. By determining how many passengers board and alight of each train at each station, the cost function of the optimization can be adjusted such that the weight on each arrival time is proportional to the number of passengers alighting from that train at that station. Weights can also be put on the variables determining whether or not a connection is maintained, with a weight proportional to the number of people missing the connection and the extra delay they incur because of that.

- Major disturbances and integration with rolling stock and personnel schedules
  Our research on railway traffic management and most research in literature has focused on minor disturbances and how to deal with those. So far very little research has been on how to handle major disruptions such as blocked tracks, power outages resulting in large parts of the network being inaccessible, or reduced velocity of all trains due to sudden (extreme) weather changes, which require major changes to the timetable and possibly implementing emergency timetables. Emergency timetables are usually available, but the hardest part may be how to get from the current situation to the emergency timetable and once the disruptions are gone how to effectively return to the normal timetable. This will require an integration of rolling stock, personnel, and timetable schedules and methods to adjust all of them at the same time. A good first step would be to extend the current methods for railway traffic management such that they can quickly determine a new emergency schedule and a way to transition to that schedule. This will require extensions of the current approaches such that they can short turn, reroute trains through the entire network, cancel trains, and introduce new trains. A way to integrate the rolling stock, personnel schedules, and timetable schedules would be to use a hierarchical or multi-level control scheme, where the new timetable is determined first, then for that schedule the rolling stock problem is solved, if it is infeasible extra constraints are added to the rescheduling problem and a new timetable is determined, then the process is repeated. Once a feasible solution has been found the personnel scheduling problem should be solved and if that cannot be solved with the new timetable, constraints are added to the railway traffic rescheduling problem and the whole process is repeated again.

- Real world application
  Future research should also be on the realization of automated railway traffic management. A great deal of research has already been published on how these systems

work in theory, but very few are ever implemented or even tested. The models need to be verified and the control algorithms tested. This will require cooperation with the railway operators and railway managers. Interfaces must be built between the real world and the computer systems, the models need to be verified and if they do not suffice, they need to be adjusted.

6.2.2 Legged locomotion

The second subject of this thesis was the modeling and control of the locomotion patterns of legged robots. An approach based on switching max-plus-linear systems was proposed, where each gait is described by a max-plus-linear system and switching between gaits can be done by changing the system matrices of the max-plus-linear system. The class of gaits that were considered were limited to gaits without an aerial phase, and the moment of gait switching was limited to the start/end of a cycle. These limitations should be tackled in future research:

- Gaits with aerial phases
  For legged locomotion a possible subject for future research would be to consider gaits with aerial phases and possibly other more general gaits. Aerial phases are already theoretically possible in the framework presented in this paper. By setting $\tau\Delta < 0$ aerial phases can be achieved, but what kind of effects this will have on gait transitions and steady state behavior still has to be determined.

- Gait switching at any given time
  Currently the gait transitions occur after a full cycle in the max-plus gait scheduler has been completed, but to be able to start switching gaits at any given moment the research should be extended to include gait switches during cycles. A possible way to do this may be to redefine the previous cycle, at the moment of transition, such that the previous cycle consists of the most recent touchdown and lift-off times of each of the legs. Then the proposed method of gait switches in this thesis can be used to perform the gait switch at any given time.

- Energy efficient locomotion
  For a legged robot to be as mobile as possible it should have an internal power supply. Such an internal power supply has a limited capacity, therefore it is important that the robot uses its energy efficiently. With that in mind a possible subject for future research would be to determine the most energy efficient gaits and gait parameters for different velocities.

6.2.3 Additional research recommendations

Finally some general recommendations for future research are given:

- Multi-modal public transportation
  When using the public transportation passengers often transfer from one means of transport to the other, e.g. from the bus to the train, or from tram to subway.

In order to provide the best service for passengers, a multi-modal model including all types of public transport should be developed such that it can take all of these transfers between different means of transportation into account. A first step can be to set up communication to notify the operators of the other modes of transportation when there are delays and to include constraints in each of the separate scheduling problems to ensure transfers to the other modes of public transportation. Canceling these transfers should be possible, but only if canceling them results in a large reduction of the total (passenger) delays.

- Baggage handling systems
  Behind the scenes on airports there are major transportation systems that move the luggage from check-in to the airport apron and from the apron to the baggage claim area. These baggage handling systems can be described in a similar fashion as railway systems, where the events are the arrivals of the different pieces of luggage at specific points in the network and discrete choices have to be made about the route the luggage takes. Since the baggage handling systems can be described as a discrete-event system similar methods as the model predictive control and distributed model predictive control methods in this thesis can be used to optimize the operation of this system. Instead of minimizing the delay in the network the goal could be maximizing the throughput of the network, or minimize the sum of total travel time, or minimize the maximum travel time.

Appendix A

Train lines and their frequencies

The train lines and their frequencies are shown in Table A.1. Each line is in both directions and the frequency is per direction.

Table A.1: Train lines and their frequencies for the 2011 timetable of the Dutch Network. InterCity (IC) trains are interregional trains and Sprinters (SP) are local trains.

Train line	Train type	Frequency
Alkmaar - Maastricht CS	IC	every 30 minutes
Arnhem - Ede-Wageningen	SP	every 60 minutes
Arnhem - Zutphen	SP	every 30 minutes
Amsterdam CS - Almere Oostvaarders	SP	every 30 minutes
Amsterdam CS - Amersfoort	SP	every 30 minutes
Amsterdam CS - Breda	SP	every 30 minutes
Amsterdam CS - Den Haag CS	IC	every 30 minutes
Amsterdam CS - Dordrecht	IC	every 60 minutes
Amsterdam CS - Haarlem	SP	every 30 minutes
Amsterdam CS - Hoofddorp	SP	every 30 minutes
Amsterdam CS - Lelystad Centrum	IC	every 30 minutes
Amsterdam CS - Rotterdam CS	IC	every 60 minutes
Amsterdam CS - Roosendaal	IC	every 60 minutes
Amsterdam CS - Uitgeest (Haarlem)	SP	every 30 minutes
Amsterdam CS - Uitgeest (Zaandam)	SP	every 30 minutes
Amsterdam CS - Vlissingen	IC	every 60 minutes
Apeldoorn - Enschede	SP	every 30 minutes
Breukelen - Rhenen	SP	every 30 minutes
Den Haag CS - Breda	SP	every 30 minutes
Den Haag CS - Enschede	IC	every 60 minutes
Den Haag CS - Groningen	IC	every 60 minutes
Den Haag CS - Gouda Goverwelle	SP	every 30 minutes
Den Haag CS - Haarlem	SP	every 30 minutes

Continued on next page

Table A.1 – continued from previous page

Train line	Train type	Frequency
Den Haag CS - Hoorn	SP	every 30 minutes
Den Haag CS - Lelystad Centrum	IC	every 30 minutes
Den Haag CS - Lelystad Centrum	SP	every 30 minutes
Den Haag CS - Utrecht CS	IC	every 30 minutes
Den Haag CS - Utrecht CS	SP	every 30 minutes
Den Haag CS - Venlo	IC	every 30 minutes
Den Haag CS - Roosendaal	SP	every 30 minutes
Den Helder - Nijmegen	IC	every 30 minutes
Enkhuizen - Amersfoort	IC	every 30 minutes
Eindhoven - Weert	SP	every 30 minutes
Eindhoven - Tilburg	SP	every 30 minutes
Groningen - Zwolle	IC	every 60 minutes
's Hertogenbosch - Nijmegen	SP	every 30 minutes
's Hertogenbosch - Deurne	SP	every 30 minutes
Hoorn - Hoofddorp	SP	every 30 minutes
Roermond - Maastricht Randwyck	SP	every 30 minutes
Roosendaal - Vlissingen	IC	every 60 minutes
Roosendaal - Zwolle	SP	every 30 minutes
Rotterdam CS - Amersfoort	IC	every 30 minutes
Rotterdam CS - Deventer	IC	every 60 minutes
Rotterdam CS - Leeuwarden	IC	every 60 minutes
Rotterdam CS - Uitgeest	SP	every 30 minutes
Schiphol - Groningen	IC	every 60 minutes
Schiphol - Eindhoven	IC	every 30 minutes
Schiphol - Enschede	IC	every 60 minutes
Schiphol - Leeuwarden	IC	every 60 minutes
Schiphol - Nijmegen	IC	every 30 minutes
Sittard - Heerlen	IC	every 30 minutes
Sittard - Heerlen	SP	every 30 minutes
Utrecht CS - Almere Centrum	IC	every 30 minutes
Utrecht CS - Breda	SP	every 30 minutes
Utrecht CS - Breukelen	SP	every 30 minutes
Utrecht CS - Hoofddorp	SP	every 30 minutes
Utrecht CS - Tiel	SP	every 30 minutes
Utrecht CS - Zwolle	SP	every 30 minutes

Bibliography

[1] S. Aoi, T. Yamashita, A. Ichikawa, and K. Tsuchiya. Hysteresis in gait transition induced by changing waist joint stiffness of a quadruped robot driven by nonlinear oscillators with phase resetting. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1915–1920, Taipei, Taiwan, 2010.

[2] Y. Asakura, T. Iryo, Y. Nakajima, and T. Kusakabe. Estimation of behavioural change of railway passengers using smart card data. *Public Transport*, 4(1):1–16, 2012.

[3] F. Baccelli, G. Cohen, G. J. O. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Wiley, New York, 1992.

[4] N. Bešinović, E. Quaglietta, and R. M. P. Goverde. A simulation-based optimization approach for the calibration of dynamic train speed profiles. *Journal of Rail Transport Planning & Management*, 3(4):126–136, 2013.

[5] J. G. Braker. Max-algebra modelling and analysis of time-dependent transportation networks. In *Proceedings of the 1st European Control Conference*, pages 1831–1836, Grenoble, France, July 1991.

[6] J. G. Braker. An extended algorithm for performance evaluation of timed event graphs. In *Proceedings of the 2nd European Control Conference*, pages 524–529, Groningen, The Netherlands, June 1993.

[7] V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, and J. Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, 2014.

[8] G. Caimi, M. Fuchsberger, M. Laumanns, and M. Lüthi. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers and Operations Research*, 39(11):2578–2593, 2012.

[9] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar. Distributed model predictive control. *Control Systems, IEEE*, 22(1):44–52, Feb 2002.

[10] F. Corman. *Real-time Railway Traffic Management: dispatching in complex, large and busy railway networks*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2010/14, Delft, The Netherlands, 2010.

[11] F. Corman and L. Meng. A review of online dynamic models and algorithms for railway traffic management. *Intelligent Transportation Systems, IEEE Transactions on*, 16(3):1274–1284, June 2015.

[12] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1):175–192, 2010.

[13] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport*, 2 (3):219–247, 2010.

[14] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):71–88, 2012.

[15] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C: Emerging Technologies*, 20(1):79–94, 2012.

[16] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research*, 44(1):146–160, 2014.

[17] R. A. Cuninghame-Green. Process synchronisation in a steelworks – A problem of feasibility. In J. Banbury and J. Maitland, editors, *Proceedings of the 2nd International Conference on Operations Research*, pages 323–328. London: English University Press, 1960.

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

[19] R. A. Cuninghame-Green. *Minimax Algebra*, volume 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Germany, 1979.

[20] C. R. Cutler and B. L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings of the Joint Automatic Control Conference*, San Francisco, California, 1980.

[21] A. D'Ariano. *Improving Real-Time Train Dispatching: Models, Algorithms and Applications*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2008/6, Delft, The Netherlands, 2008.

[22] A. D'Ariano and M. Pranzo. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics*, 9(1):63–84, 2009.

[23] A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.

[24] S. Daun-Gruhn and T. I. Toth. An inter-segmental network model and its use in elucidating gait-switches in the stick insect. *Journal of Computational Neuroscience*, 31(1):43–60, 2011.

[25] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, 2001.

[26] B. De Schutter and T. van den Boom. MPC for discrete-event systems with soft and hard synchronisation constraints. *International Journal of Control*, 76(1):82–94, Jan. 2003.

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

[28] P. de Waal, A. Overkamp, and J. H. van Schuppen. Control of railway traffic on a single line. In *Proceedings of the European Control Conference (ECC'97)*, Brussels, Belgium, paper 230, July 1997.

[29] W. Dunbar. Distributed receding horizon control of dynamically coupled nonlinear systems. *IEEE Transactions on Automatic Control*, 52(7):1249–1263, July 2007.

[30] W. Fang, S. Yang, and X. Yao. A survey on problem models and solution approaches to rescheduling in railway networks. *Intelligent Transportation Systems, IEEE Transactions on*, pages 1–20, 2015.

[31] M. Farina and R. Scattolini. Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems. *Automatica*, 48 (6):1088–1096, 2012.

[32] C. E. Garcí, D. M. Prett, and M. Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):335–348, 1989.

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

[34] B. Giffler. Schedule algebra: A progress report. *Naval Research Logistics Quarterly*, 15(2):255–280, June 1968.

[35] GLPK. Gnu Linear Programming Kit, 2014. URL http://www.gnu.org/software/glpk/.

[36] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 3785–3790, April 2004.

[37] R. M. P. Goverde. *Punctuality of Railway Operations and Timetable Stability Analysis*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2005/10, Delft, The Netherlands, 2005.

[38] R. M. P. Goverde. Railway timetable stability analysis using max-plus system theory. *Transportation Research Part B: Methodological*, 41(2):179–201, 2007.

[39] R. M. P. Goverde. A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3):269–287, 2010.

[40] Gurobi. Gurobi optimizer reference manual, 2014. URL http://www.gurobi.com.

[41] G. C. Haynes and A. A. Rizzi. Gaits and gait transitions for legged robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1117–1122, Orlando, Florida, USA, 2006.

[42] G. C. Haynes and A. A. Rizzi. Gait regulation and feedback on a robotic climbing hexapod. In *Robotics: Science and Systems*, Philadelphia, USA, 2006.

[43] G. C. Haynes, F. R. Cohen, and D. E. Koditschek. Gait transitions for quasi-static hexapedal locomotion on level ground. In *Proceedings of the International Symposium of Robotics Research*, pages 105–121, Lucerne, Switzerland, 2009.

[44] B. Heidergott and R. Vries. Towards a (max,+) control theory for public transportation networks. *Discrete Event Dynamic Systems*, 11(4):371–398, 2001.

[45] B. Heidergott, G. J. Olsder, and J. van der Woude. *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2006.

[46] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, volume 2, pages 1321–1326, May 1998.

[47] P. Holmes, R. Full, D. Koditschek, and J. Guckenheimer. The dynamics of legged locomotion: models, analyses, and challenges. *SIAM Review*, 48(2):207–304, 2006.

[48] A. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[49] S. Inagaki, H. Yuasa, and T. Arai. CPG model for autonomous decentralized multi-legged robot system - generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(3–4):171–179, 2003.

[50] S. Inagaki, H. Yuasa, T. Suzuki, and T. Arai. Wave CPG model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118–126, 2006.

[51] S. Kanai, K. Shiina, S. Harada, and N. Tomii. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. *Journal of Rail Transport Planning & Management*, 1(1):25–37, 2011. Robust Modelling of Capacity, Delays and Rescheduling in Regional Networks.

[52] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *Proceedings of 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1083–1090, April 2004.

[53] P. Kecman. *Models for Predictive Railway Traffic Management*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2014/5, Delft, The Netherlands, 2014.

[54] P. Kecman and R. Goverde. Process mining of train describer event data and automatic conflict identification. In C. Brebbia, N. Tomii, J. Mera, B. Ning, and P. Tzieropoulos, editors, *WIT Transactions on The Built Environment, Computers in Railways XIII*, volume 127, pages 227–238. WIT press, Southampton, United Kingdom, 2012.

[55] P. Kecman and R. Goverde. Adaptive, data-driven, online prediction of train event times. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC2013)*, pages 803–808, Oct 2013.

[56] P. Kecman, F. Corman, A. D'Ariano, and R. M. Goverde. Rescheduling models for railway traffic management in large-scale networks. *Public Transport*, 5(1-2):95–123, 2013.

[57] B. Kersbergen, J. Rudan, T. van den Boom, and B. De Schutter. Towards railway traffic management using switching max-plus-linear systems. *Discrete Event Dynamic Systems*, pages 1–41, 2014.

[58] B. Kersbergen, T. J. J. van den Boom, and B. De Schutter. Distributed model predictive control for rescheduling of railway traffic. In *Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC2014)*, pages 2732–2737, China, October 2014.

[59] B. Kersbergen, T. J. J. van den Boom, and B. De Schutter. Improved distributed model predictive control for rescheduling of railway traffic by manipulation of the cost functions. In *Proceedings of the 6th International Seminar on Railway Operations Modelling and Analysis (RailTokyo)*, pages 25:1–13, Japan, March 2015.

[60] E. Klavins and D. Koditschek. Phase regulation of decentralized cyclic robotic systems. *International Journal of Robotics Research*, 21(3):257–275, 2002.

[61] T. Kusakabe, T. Iryo, and Y. Asakura. Estimation method for railway passengers' train choice behavior with smart card transaction data. *Transportation*, 37(5):731–749, 2010.

[62] B. Li, Y. Li, and X. Rong. Gait generation and transitions of quadruped robot based on wilson-cowan weakly neural networks. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 19–24, Tianjin, China, 2010.

[63] T. Li, Y.-T. Su, S.-H. Liu, J.-J. Hu, and C.-C. Chen. Dynamic balance control for biped robot walking using sensor fusion, kalman filter, and fuzzy logic. *IEEE Transactions on Industrial Electronics*, 59(11):4394–4408, Nov 2012.

[64] G. A. D. Lopes. Abstractions for legged locomotion. In K. Kozlowski, M. O. Tokhi, and G. S. Virk, editors, *Mobile Service Robotics*. World Scientific, 2014.

[65] G. A. D. Lopes, R. Babuška, B. De Schutter, and A. J. J. van den Boom. Switching max-plus models for legged locomotion. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 221–226, Dec 2009.

[66] G. A. D. Lopes, T. J. J. van den Boom, B. De Schutter, and R. Babuška. Modeling and control of legged locomotion via switching max-plus systems. In *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, pages 392–397, Berlin, Germany, Aug.–Sept. 2010.

[67] G. A. D. Lopes, B. Kersbergen, T. J. J. van den Boom, B. De Schutter, and R. Babuška. Modeling and control of legged locomotion via switching max-plus models. *IEEE Transactions on Robotics*, 30(3):652–665, June 2014.

[68] G. A. D. Lopes, B. Kersbergen, B. De Schutter, T. van den Boom, and R. Babuška. Synchronization of a class of cyclic discrete-event systems describing legged locomotion. *Discrete Event Dynamic Systems*, pages 1–37, 2015.

[69] J. M. Maestre and R. R. Negenborn. *Distributed Model Predictive Control Made Easy*. Springer Publishing Company, Incorporated, 2013.

[70] R. Minciardi, M. Paolucci, and R. Pesenti. Generating optimal schedules for an underground railway line. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 4082–4085, New Orleans, Louisiana, Dec. 1995.

[71] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4–5):667–682, 1999.

[72] H. Nagashino, Y. Nomura, and Y. Kinouchi. A neural network model for quadruped gait generation and transitions. *Neurocomputing*, 38–40(0):1469–1475, 2001.

[73] S. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[74] M. Raibert, K. Blankespoor, R. P. Gabriel Nelson, and the BigDog Team. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th IFAC World Congress*, pages 10822–10825, July 2008.

[75] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[76] A. Richards and J. P. How. Robust distributed model predictive control. *International Journal of Control*, 80(9):1517–1531, 2007.

[77] J. Rodriguez. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, 41(2):231–245, 2007. Advanced Modelling of Train Operations in Stations and Networks.

[78] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: system overview and integration. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2478–2483, October 2002.

[79] C. P. Santos and V. Matos. Gait transition and modulation in a quadruped robot: A brainstem-like modulation approach. *Robotics and Autonomous Systems*, 59(9): 620–634, 2011.

[80] U. Saranli, M. Buehler, and D. E. Koditschek. Rhex: a simple and highly mobile hexapod robot. *International Journal of Robotics Research*, 20(7):616–631, 2001.

[81] TOMLAB. Tomlab optimization environment, 2014. URL http://tomopt.com/tomlab/.

[82] J. Törnquist. Railway traffic disturbance management – an experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. *Transportation Research Part A: Policy and Practice*, 41(3):249–266, 2007.

[83] J. Törnquist and J. A. Persson. N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362, 2007.

[84] J. Törnquist-Krasemann. Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*, 20(1):62–78, 2012.

[85] T. J. J. van den Boom and B. De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14 (10):1199–1211, Oct. 2006.

[86] T. J. J. van den Boom, N. Weiss, W. Leune, R. M. P. Goverde, and B. De Schutter. A permutation-based algorithm to optimally reschedule trains in a railway traffic network. In *Proceedings of the 18th IFAC World Congress*, pages 9537–9542, Milan, Italy, Aug.–Sept. 2011.

[87] E. van der Hurk, L. Kroon, G. Maroti, and P. Vervest. Deduction of passengers' route choices from smart card data. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1):430–440, Feb 2015.

[88] M. Vukobratovic and B. Borovac. Zero-moment point – thirty five years of its life. *International Journal Of Humanoid Robotics*, 01(01):157–173, 2004.

[89] Y. Wang. *Optimal Trajectory Planning and Train Scheduling for Railway Systems*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2014/7, Delft, The Netherlands, 2014.

[90] Y. Wang, B. D. Schutter, T. J. van den Boom, and B. Ning. Optimal trajectory planning for trains - a pseudospectral method and a mixed integer linear programming approach. *Transportation Research Part C: Emerging Technologies*, 29(0):97–114, 2013.

[91] Y. Wang, B. D. Schutter, T. J. van den Boom, and B. Ning. Optimal trajectory planning for trains under fixed and moving signaling systems using mixed integer linear programming. *Control Engineering Practice*, 22(0):44–56, 2014.

[92] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–297, 1951.

[93] J. Weingarten, R. Groff, and D. Koditschek. A framework for the coordination of legged robot gaits. In *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, pages 679–686, Singapore, 2004.

[94] J. Yuan. *Stochastic modelling of train delays and delay propagation in stations*. PhD thesis, Delft University of Technology, TRAIL Thesis Series T2006/6, Delft, The Netherlands, 2006.

[95] X. Zhang, H. Zheng, and L. Chen. Gait transition for a quadrupedal robot by replacing the gait matrix of a central pattern generator model. *Advanced Robotics*, 20(7):849–866, 2006.

[96] C. Zhou and Q. Meng. Dynamic balance of a biped robot using fuzzy reinforcement learning agents. *Fuzzy Sets and Systems*, 134(1):169–187, 2003. Fuzzy Set Techniques for Intelligent Robotic Systems.

TRAIL Thesis Series publications

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 100 titles, see the TRAIL website: www.rsTRAIL.nl. The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Kersbergen, B., *Modeling and Control of Switching Max-Plus-Linear Systems: Rescheduling of railway traffic and changing gaits in legged locomotion*, T2015/16, October 2015, TRAIL Thesis Series, the Netherlands

Brands, T., *Multi-Objective Optimisation of Multimodal Passenger Transportation Networks*, T2015/15, October 2015, TRAIL Thesis Series, the Netherlands

Ardiç, Özgül, *Road Pricing Policy Process: The interplay between policy actors, the media and public*, T2015/14, September 2015, TRAIL Thesis Series, the Netherlands

Xin, J., *Control and Coordination for Automated Container Terminals*, T2015/13, September 2015, TRAIL Thesis Series, the Netherlands

Anand, N., *An Agent Based Modelling Approach for Multi-Stakeholder Analysis of City Logistics Solutions*, T2015/12, September 2015, TRAIL Thesis Series, the Netherlands

Hurk, E. van der, *Passengers, Information, and Disruptions*, T2015/11, June 2015, TRAIL Thesis Series, the Netherlands

Davydenko, I., *Logistics Chains in Freight Transport Modelling*, T2015/10, May 2015, TRAIL Thesis Series, the Netherlands

Schakel, W., *Development, Simulation and Evaluation of In-car Advice on Headway, Speed and Lane*, T2015/9, May 2015, TRAIL Thesis Series, the Netherlands

Dorsser, J.C.M. van, *Very Long Term Development of the Dutch Inland Waterway Transport System: Policy analysis, transport projections, shipping scenarios, and a new perspective on economic growth and future discounting*, T2015/8, May 2015, TRAIL Thesis Series, the Netherlands

Hajiahmadi, M., *Optimal and Robust Switching Control Strategies: Theory, and applications in traffic management*, T2015/7, April 2015, TRAIL Thesis Series, the Netherlands

Wang, Y., *On-line Distributed Prediction and Control for a Large-scale Traffic Network*, T2015/6, March 2015, TRAIL Thesis Series, the Netherlands

Vreeswijk, J.D., *The Dynamics of User Perception, Decision Making and Route Choice*, T2015/5, February 2015, TRAIL Thesis Series, the Netherlands

Lu, R., *The Effects of Information and Communication Technologies on Accessibility*,

T2015/4, February 2015, TRAIL Thesis Series, the Netherlands

Ramos, G. de, *Dynamic Route Choice Modelling of the Effects of Travel Information using RP Data*, T2015/3, February 2015, TRAIL Thesis Series, the Netherlands

Sierzchula, W.S., *Development and Early Adoption of Electric Vehicles: Understanding the tempest*, T2015/2, January 2015, TRAIL Thesis Series, the Netherlands

Vianen, T. van, *Simulation-integrated Design of Dry Bulk Terminals*, T2015/1, January 2015, TRAIL Thesis Series, the Netherlands

Risto, M., *Cooperative In-Vehicle Advice: A study into drivers' ability and willingness to follow tactical driver advice*, T2014/10, December 2014, TRAIL Thesis Series, the Netherlands

Djukic, T., *Dynamic OD Demand Estimation and Prediction for Dynamic Traffic Management*, T2014/9, November 2014, TRAIL Thesis Series, the Netherlands

Chen, C., *Task Complexity and Time Pressure: Impacts on activity-travel choices*, T2014/8, November 2014, TRAIL Thesis Series, the Netherlands

Wang, Y., *Optimal Trajectory Planning and Train Scheduling for Railway Systems*, T2014/7, November 2014, TRAIL Thesis Series, the Netherlands

Wang, M., *Generic Model Predictive Control Framework for Advanced Driver Assistance Systems*, T2014/6, October 2014, TRAIL Thesis Series, the Netherlands

Kecman, P., *Models for Predictive Railway Traffic Management*, T2014/5, October 2014, TRAIL Thesis Series, the Netherlands

Davarynejad, M., *Deploying Evolutionary Metaheuristics for Global Optimization*, T2014/4, June 2014, TRAIL Thesis Series, the Netherlands

Li, J., *Characteristics of Chinese Driver Behavior*, T2014/3, June 2014, TRAIL Thesis Series, the Netherlands

Mouter, N., *Cost-Benefit Analysis in Practice: A study of the way Cost-Benefit Analysis is perceived by key actors in the Dutch appraisal practice for spatial-infrastructure projects*, T2014/2, June 2014, TRAIL Thesis Series, the Netherlands

Ohazulike, A., *Road Pricing mechanism: A game theoretic and multi-level approach*, T2014/1, January 2014, TRAIL Thesis Series, the Netherlands

Cranenburgh, S. van, *Vacation Travel Behaviour in a Very Different Future*, T2013/12, November 2013, TRAIL Thesis Series, the Netherlands

Samsura, D.A.A., *Games and the City: Applying game-theoretical approaches to land and property development analysis*, T2013/11, November 2013, TRAIL Thesis Series, the Netherlands

Huijts, N., *Sustainable Energy Technology Acceptance: A psychological perspective*, T2013/10, September 2013, TRAIL Thesis Series, the Netherlands

Samenvatting

De werking van veel systemen kan beschreven worden aan de hand van de timing van gebeurtenissen, zoals bij vliegvelden, waar de vliegtuigen landen en vertrekken op specifieke tijden, bij fabricage processen en chemische processen, waar materialen verschillende processen ondergaan in een specifieke volgorde om zodoende het gewenste eindproduct te verkrijgen, bij spoorverkeer, waar de treinen vertrekken en aankomen volgens de dienstregeling, en bij de voortbeweging van dieren met poten, waar de poten worden opgetild en neergezet in een bepaalde volgorde. Deze systemen kunnen beschouwd worden als zogenaamde systemen met discrete gebeurtenissen waar de gebeurtenissen (aankomsten, vertrekken, de start en het einde van een proces, het optillen en neerzetten van poten) met elkaar zijn verbonden door middel van voorwaarden. Wanneer het systeem beschreven kan worden door vergelijkingen die "lineair" zijn in de max-plus algebra, welke maximalisatie en optelling als basis operatoren heeft, dan is wordt het systeem een max-plus-lineair systeem genoemd.

In veel van deze systemen kan het nodig zijn om de volgorde van de gebeurtenissen aan te passen door veranderingen in de omstandigheden, of de voorwaarden, of door onvoorziene omstandigheden. Voor elke mogelijke volgorde van gebeurtenissen is een andere max-plus-lineaire systeembeschrijving nodig. Dit soort systemen, dat de volgorde van gebeurtenissen kan aanpassen worden schakelende max-plus-lineaire (SMPL) systemen genoemd.

In dit proefschrift beschouwen we twee toepassingen van SMPL systemen. De eerste toepassing van SMPL systemen modelleert de netwerken met treinverkeer en wordt gebruikt voor het on-line herplannen van het treinverkeer in het geval van vertragingen. In dit proefschrift wordt een macroscopisch model voor het netwerk met treinverkeer gepresenteerd dat de gevolgen van verschillende regel acties kan modeleren zoals: het veranderen van de volgorde van treinen, het verbreken van verbindingen, het wisselen van treinen over parallelle sporen tussen stations, en het annuleren van koppel opdrachten tussen treinen. Voor elke reeks van regel acties wordt de nieuwe volgorde en het tijdstip van de gebeurtenissen bepaald. De structuur van de systeem matrices van het SMPL systeem is geanalyseerd en het is aangetoond hoe de structuur gebruikt kan worden om het systeem van zijn impliciete naar zijn expliciete vorm geconverteerd kan worden. Er is een reductie methode ontwikkeld, die profiteert van de structuur van het expliciete model zodat het de complexiteit van het expliciete model significant kan verminderen, door het aantal, dat de gebeurtenissen verbinden aanzienlijk te verminderen.

Om het probleem van het on-line herplannen voor het netwerk met treinverkeer op

te lossen wordt een globaal model-gebaseerde voorspellende regeltechniek aanpak en vier gedistribueerde model-gebaseerde voorspellende regeltechniek aanpakken voorgesteld. Op discrete ogenblikken in de tijd moeten de regelaars een optimalisatie probleem oplossen dat geschreven kan worden als een *mixed integer* lineair programmeer probleem. Al deze aanpakken zijn uitgebreid getest op een model van het Nederlandse spoorwegnet met de treinen van de Nederlandse Spoorwegen voor verschillende lengtes van de voorspellingshorizon. Aan de hand van deze casus kunnen we concluderen dat de gedistribueerde aanpakken veel minder tijd nodig hebben om een oplossing te vinden voor het mixed integer lineair programmeer probleem, terwijl gemiddeld genomen de oplossing gevonden met de gedistribueerde aanpakken maar iets minder optimaal is dan die van de globale aanpak qua reductie van vertragingen.

De tweede toepassing van SMPL systemen modelleert de voortbeweging van een robot door middel van het gebruik van poten voor verschillende gangen. In dit proefschrift wordt het cyclische stabiele gedrag van het max-plus-lineaire systeem dat de gangen beschrijft, en de overgang naar het cyclische stabiele gedrag, geanalyseerd. We hebben bewezen dat, onder milde voorwaarden, de eigenvector uniek is (tot aan een schaalfactor). Dit werd bewezen door aan te tonen dat de kritieke graaf van de systeem matrix bestaat uit één sterk samenhangende subgraaf.

Nu het cyclische stabiele gedrag uniek gedefinieerd is voor alle gangen, konden we schakelingen tussen gangen bepalen die de variatie in de grond tijd tussen de poten minimaliseren. Tevens hebben we een methode ontwikkeld dat de vlieg tijden van de poten tijdens de schakelingen tussen gangen aanpast om er voor te zorgen dat de grond tijd van alle poten hetzelfde is en om de schakeling te versnellen. Als laatste hebben we een methode ontwikkeld om de robot met een constant tempo te laten versnellen (of afremmen) door de functie van de tijd van de robot te manipuleren.

Bart Kersbergen

Summary

The operation of many systems can be described by the timing of events, such as airports, where planes arrive and depart at specific times, manufacturing and chemical processes, where materials need to undergo several processes in a specific order to obtain the desired end product, railway traffic, where trains depart and arrive according to a timetable, and the locomotion of legged animals, where the legs lift off and touch down in order. These systems can be considered as discrete-event systems where the events (arrivals, departures, the start and end of a process, touch down, and lift off of legs) are connected to each other through constraints. When the system behavior can be described by equations that are "linear" in the max-plus algebra, which has maximization and addition as its basic operations, the system is called a max-plus-linear system.

In many of these systems the order of the events may need to be changed due to changes in the conditions, or the requirements, or unforeseen consequences. For each possible order of events a different max-plus-linear system description is needed. Such systems that can change the order of events are called switching max-plus-linear systems. In this thesis we describe methods for the modeling and control of two types of switching max-plus-linear (SMPL) systems.

The first application of SMPL systems models the railway traffic networks and is used for on-line rescheduling of railway traffic in the case of delays. In this thesis a macroscopic model for the railway traffic network is presented that can model the effects on the railway traffic of several control actions: changing the train orders, breaking connections, switching trains over parallel tracks between stations, and canceling coupling orders for trains. For every set of control actions the new event order and times are determined. The structure of the system matrices of the SMPL system is analyzed and it is shown how the structure can be used to convert the model from its implicit into its explicit form. A reduction method is developed that takes advantage of the explicit model structure to significantly reduce the complexity of the model, in the sense that the number of constraints connecting the events is reduced considerably.

In order to solve the on-line rescheduling problem for a railway traffic network a global model predictive control (MPC) approach and four distributed model predictive control (DMPC) approaches are proposed. At discrete time instants the controllers have to solve an optimization problem that can be written as a mixed integer linear programming problem. All of these approaches are extensively tested on a model of the Dutch railway traffic network for various lengths of the prediction horizon. From this case study we can conclude that the DMPC approaches require much less time to compute a solution to

the given mixed integer linear programming problem, while on average the solution found with the DMPC approaches is only a little less optimal than the one of the centralized MPC in terms of delay reduction.

The second type of SMPL system models legged locomotion for different gaits. In this thesis the steady state cyclic behavior of the max-plus-linear systems describing the gaits, and the transition to the steady state cyclic behavior, are analyzed. It has been proven that, given mild conditions, the eigenvector is unique (up to a scaling factor). This was done by showing that the critical graph of the system matrix consists of one strongly connected subgraph.

With the steady state behavior uniquely defined for all gaits we were able to determine gait switches that minimized the variation in ground time between the legs. Furthermore a method was developed that manipulated the flight times of the legs during gait switches to ensure the ground time of all legs is the same and to speed up the gait switch. Finally a method was developed to allow the robot to accelerate (or decelerate) at a constant pace by adjusting the timing function of the robot.

Bart Kersbergen

About the author

Bart Kersbergen was born on September 4, 1987 in Nieuwegein, the Netherland. He obtained his B.Sc. degree in Electrical Engineering in 2009 from the Delft University of Technology. In 2011 Bart received his M.Sc. degree in Systems and Control from the Delft University of Technology in 2011. In August 2011 he joined the Delft Center for Systems and Control as a Ph.D. candidate. As a Ph.D. candidate Bart worked on the STW project: "Model-Predictive Railway Traffic Management" under the supervision of dr.ir. Ton van den Boom and prof.dr.ir. Bart De Schutter. His research has been focused on the development of model predictive control methods for the on-line management of railway traffic and optimal gait switching for legged robots. Bart Kersbergen followed several courses from graduate schools of the Landelijk Netwerk Mathematische Besliskunde (LNMB) and the Dutch Institute of Systems and Control (DISC). In 2013 he obtained the course certificate of DISC.

Besides his work as a Ph.D. candidate Bart has also been actively involved in the management of the local volleyball club in Benschop, as the secretary of the board from 2008 till 2014. Together with the other members of the board he was responsible for ensuring the organization of the club was operating efficiently, that there was a friendly and sporty atmosphere and that the teams had the tools they needed to perform well.