Scheduling Automated Guided Vehicles in Container Terminals Using Max-plus-linear Systems

Anne-Michelle Vos

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Scheduling Automated Guided Vehicles in Container Terminals Using Max-plus-linear Systems

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at
Delft University of Technology

Anne-Michelle Vos

July 21, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE)
Delft University of Technology

The work in this thesis was supported by TBA. Their cooperation is hereby gratefully acknowledged.

Abstract

The container seaport industry is highly competitive and transports cargo that exceeds a global trade value of US$ 4 trillion. Time efficiency is essential in container terminals of sea ports, where the total throughput can rate up to 32.5 million TEU a year. Reducing the makespan (time needed to unload and reload a ship) can therefore lead to enormous savings, even if the margin is a few per cent. The scheduling of AGVs influences this makespan. Nowadays, the scheduling of AGVs is performed by a basic dispatching method, providing an opportunity for improvement. The researched methods lack either computation speed, robustness or both. Hence the goal of this thesis is to design a robust scheduling method that is able to solve the AGV dispatching problem on-line.

A promising approach to scheduling problems is that of max-plus-linear (MPL) systems. MPL systems proved efficient in various applications, such as in railway timetable scheduling, legged locomotion or paper scheduling in printers. Using MPL models is convenient, since non-linear systems will be linear in the max-plus algebra *without compensating on the dynamics* of the system. Besides considering the dynamics of the system, it is also possible to take advantage of fast LP solvers. These solvers determine the dispatching of the AGVs by selecting different routes for the vehicles, which are all described in advance. The route selection is determined using max-plus binary control variables, turning the AGV dispatching problem into a MILP problem.

To alleviate the computational burden, model predictive scheduling (MPS) is introduced. This method uses the model to make an optimal schedule prediction, to subsequently implement only the first few control (scheduling) actions. When the schedule is recalculated, new state information can be considered and a recourse of action is regularly taken due to the horizon shift and perturbed nature of the system.

To account for the perturbations in the system, a Monte Carlo algorithm is implemented. This relatively uncomplicated method penalizes choices that may seem favorable when nominal performance of the resources is assumed, but appear disadvantageous when noise is added to the system.

Table of Contents

List of Figures

List of Tables

Preface & Acknowledgments

The document at hand is the final report of my Master's thesis, required to obtain a Master's degree in Control Engineering. The selection of the subject for my thesis was quite straightforward. As a teenager I already decided the fastest possible (relay) arrangements at competitive swimming galas, whereas now my interest has shifted to more large-scale operations. I love how existing systems contain dynamics at every operational level and how, by a good understanding of the system and the right mathematical knowledge, there is most often still room for efficiency and reliability improvement. Ton van den Boom was therefore the designated person to approach after he gave a short presentation to the new master students about the scheduling and optimization of train timetables. After our first talk on the research area of min-max-plus algebra and its relative new applications in many fields, I found myself excited to learn more about this subject. I believe Ton catches the interest of many students for his research field with his unwearying enthusiasm. He proposed the issue of scheduling automated guided vehicles in container terminals about the same time TBA had an opening for an internship on this particular topic. TBA is a fast growing international company that solves logistic problems, focusing on the cost efficiency and productivity of container terminals using simulation, emulation and by designing software. I am grateful I got the chance to seize this opportunity.

I am thankful for my two excellent supervisors, Ton van den Boom and Mernout Burger.
Ton, thank you for the wisdom you provided in the max-plus algebra and the guidance through this process. I always felt supported and could drop by when questions troubled my mind.
Mernout, I am indulged by your thirst for knowledge, which is truly admirable. Your practical view and knowledge were a tremendous help to complete my Master's thesis.

Lastly, I would like to thank my parents and Bas for their support. Bas, thank you for not taking everything too serious when I needed some distraction. Luckely you were always there to lift that rock you like to refer to so much. Mam, pap, bedankt voor al die jaren van steun en toeverlaat, jullie zorg, liefde en betrokkenheid. Dank voor zo veel meer dan woorden toelaten, ze schieten werkelijk tekort (voor deze ene keer!).

Delft, University of Technology — Anne-Michelle Vos
July 21, 2015

"What is worth the prize is always worth the fight."

— *Chad Kroeger*

Chapter 1

Problem Description

Automation proved to give cheaper and safer operations, less emissions and better infrastructure utilizations. Small efficiency improvements express themselves in huge advantages when the transshipment and throughput of containers is considered. Container-based liner service accounts for approximately 60% of the total value of shipments [1]. The total throughput of the container terminal in Rotterdam in 2012 is estimated on 11.9 million Twenty feet Equivalent Unit (TEU), of which approximately 40% is transshipped. In the worlds biggest container terminal (Shanghai) the total throughput is estimated on 32.5 million TEU in the year 2012, with a transshipment rate of approximately 82% [2]. These numbers emphasize the importance of container handling. The increasing number of container shipments causes higher demands on the seaport container terminals, container logistics and management, as well as on technical equipment [3]. This development results in an increasing competition between seaports, especially between geographically close ones. It is therefore important to optimize success factors in container seaports, in particular the time in port for ships (influenced by the makespan) combined with low rates for loading and discharging. Rapid turnover of containers is thus a crucial competitive advantage.

Container turnover involves discharging and loading operations. During discharging operations, the containers in a container ship are unloaded from the vessel and stacked in the storage yard. During loading operations, the containers are retrieved from the storage yard and are loaded onto the ship [4]. The (un)loading of the container ship is performed by quay cranes (QCs) and the storing and retrieving in the yard is done by yard cranes (YCs). For the transportation of containers between the quay and yard, horizontal transport means are used. One of the handling equipment that is applicable for this, is the automated guided vehicle (AGV).

Automated Guided Vehicles (AGVs) are an important part of an automated container terminal. Nowadays they are widely incorporated in sea port container handling sites [5]. Using AGVs has among others the advantages that they are accurate ($\pm$25mm) [6], their routes remain unchanged and are therefore predictable, they operate without human intervention,

they are integrated with a traffic management system that guarantees no collisions and they are flexible, since their route can be modified in a matter of seconds. However, the scheduling and routing of the horizontal transport means (including AGVs) is also a difficult optimization problem, which has a large influence on the makespan. Since container ships sail on a schedule, the cargo stays at the harbor when the reloading of the vessel is not done in time. Therefore both the harbor and their clients are of benefit when the AGV scheduling and routing is optimized.

In AGV scheduling, the departure times and arrival times of the AGVs are to be determined. When considering the definition of the optimal performance, the most evident criteria would be to reduce the makespan (last QC movement serving a particular ship). This is impeded by the uncertain handling times and AGV travel times in the container terminal.

Motivation

The competitive environment in container terminal operations forces advising companies to design better scheduling algorithms. TBA indicates that a *robust schedule* is of great importance to practical applications, due to the highly uncertain handling times within the container terminal. This also became clear in the preceding literature study of this thesis. Stochastic switching max-plus-linear systems and model predictive scheduling can provide a robust solution to the scheduling problem, creating an opportunity to design a flexible program that will schedule AGVs in a more efficient and robust way. Furthermore, the company emphasize the importance of a *fast computation time* of an algorithm that only needs little resources such as serves to suppress the operational costs.

1-1 Research questions

The aim is to find a on-line feasible solution of the AGV dispatching problem that minimizes congestions at the cranes and finds a (near) optimal solution for the minimization of the makespan. The algorithm should resolve congestions that may occur due to possible technical failures of the equipments and it should be able to deal with the major uncertain handling times within the port. Considering the introduction above and taking the goal into account, a research question is formulated:

What are the current possibilities and future perspectives for scheduling AGVs in an automated container terminal using max-plus-linear systems?

The max-plus algebra has been proven to deal efficiently with scheduling problems [7–10]. With the fairly newly applied algebra, it is possible to model a non-linear system in a max-plus-linear framework. The demand for an on-line scheduling method can be satisfied by introducing model predictive scheduling. Finally, the uncertain handling times will require to introduce stochastic variables, which will result in a schedule that is based on a model predictive, stochastic switching max-plus-linear system.

1-2 Outline of this thesis

The outline of this thesis is as follows:

First, general information will be provided on container terminals, max-plus algebra and on max-plus-linear (MPL) systems. Chapter 2 will give a general overview of the most important features of a container terminal and Chapter 3 introduces the max-plus algebra and (stochastic switching) max-plus-linear systems.

The following part of this Master's thesis will explain the model, optimization and the different add-ons to enhance the performance. In Chapter 4 an initial model of the system is designed and explained. Once the system is described, the optimization can be performed. In Chapter 5, a MILP and Benders' decomposition approach are taken and the results analyzed. Hereafter, the problem will be defined as a model predictive scheduling (MPS) problem, which is again recast into a MILP problem formulation and evaluated (Chapter 6). Finally in Chapter 7, the perturbed system will be described in and solved using a Monte Carlo algorithm.

Chapter 8 discusses the most important findings by reviewing the different models shortly. Additionally, the final conclusions of this Master's thesis are presented and recommendations for future research are done.

Chapter 2

The Container Terminal

In this chapter, a brief introduction to the container terminal is provided. Possible layouts of container terminals are considered and a short overview of a few common handling equipment will be presented. Finally, some dimensions and typical handling times regarding the terminal are mentioned, such that it is clear how fast the designed scheduling method should be able to schedule in an on-line application.

2-1 Layout and dynamics

The system of the container terminal will be partly modeled. In the AGV dispatching problem only the quay area up until the yard is relevant. This means that the hinterland operations are disregarded. The container ships dock at the quay area where quay cranes (QCs) unload the cargo. While doing so, the QCs can move along the quay. The automated guided vehicles (AGVs) travel also along the quay underneath the QCs to pick up the containers. Hereafter AGVs transport the containers to the yard [11]. When a container is transported from the quay side to the yard, it is named a discharge job (or ship-to-yard (S-Y) cycle). It is however also possible that the container is retrieved from the yard and loaded onto the vessel. This is then called a loading job (or yard-to-ship (Y-S) cycle). The path that an AGV follows, is predefined and will not be examined any further. When an AGV arrives at the yard, it takes position at a transfer point or lane to get (un)loaded by a yard crane (YC). This crane will eventually store the container in a stack in the yard.

Most yards have one of the two typical layouts, which are either blocks that run parallel to the quay or blocks that are perpendicular to the quay as can be seen in Figure 2-1 [3]. In this thesis, the perpendicular layout with transfer points will be assumed when modeling the system. However, it should be kept in mind that other layouts are not uncommon in container terminals.

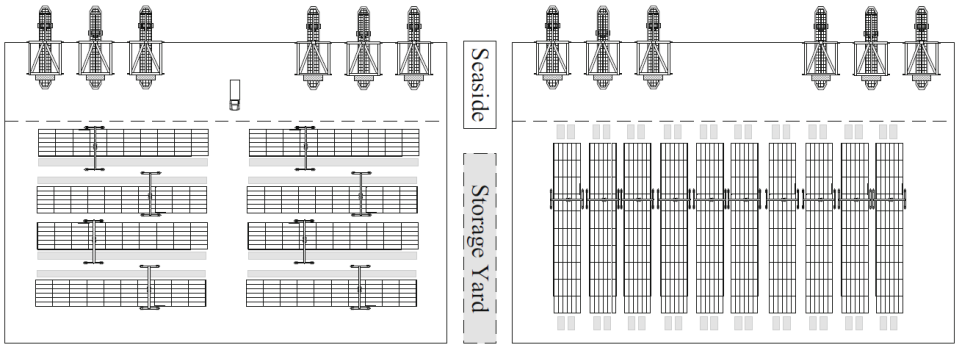

Figure 2-1: Parallel (left) and perpendicular (right) layout of seaport container terminals.

2-2 Handling equipment

Different kinds of handling equipment can be used to transport the containers. Part of the operations are often automated in large container terminals, but this is not a necessity. The handling equipment that is assumed in this work will be discussed.

2-2-1 Quay side

At the quay side, the container ships are loaded and unloaded by QCs, depicted in Figure 2-2 [12]. These cranes serve deep-sea vessels (capacity up to 13.000 TEU), but also the smaller feeder vessels (capacity 100 TEU to 1200 TEU) [2]. There are different types of quay cranes [13]: single-trolley cranes and dual-trolley cranes. In this work it is assumed that the QCs are single trolley cranes, which move the containers between ship and shore [14]. Another assumption is that these cranes place the containers on the ship or directly on an AGV and that they only move one 20 or 40 feet (1 TEU or 2 TEU) container.

Figure 2-2: QCs serve vessels at shore.

2-2-2 Transport technologies

Figure 2-3: A loaded automated guided vehicle (AGV) transporting a two TEU container.

The AGV, shown in Figure 2-3 [15], is chosen as the horizontal transport mean. The AGV is a passive transport mode, since they need to be served by a crane and are not able to lift containers themselves. It is not possible to decouple the processes of the AGV and QC.

The automated guided vehicle

An AGV system exists of two main subsystems: hardware and software [16]. The hardware includes the physical components, e.g. the controllers, sensors, guidance devices, etc.

The software embodies the algorithms for systematically managing the hardware resources to obtain the highest efficiency. It is hard to provide conclusive specifications of an AGV, since they neither contain the same hardware nor software. Usually, an AGV is diesel-electric or battery-powered and can carry either one 40/45 feet container of maximal 40 tonnes or two 20 feet containers with a combined weight of 70 tonnes. The dimensions of an AGV are approximately 15 by 3m. The forward as well as the reverse speed is maximal 6m/s, while in a maximum curve this speed is 3m/s [6]. Acceleration and deceleration are both approximately 0.5m/s^2. A last important point to mention is the (possible) tank capacity of 1400 liters and the fuel consumption of 8l/hour, providing an operation radius of 175 hours. Since the in-port time of a vessel is usually around 24 hours, it is admissible to ignore refueling of the vehicles in simulations.

2-2-3 Yard side

Depending on the layout of the terminal, different kinds of YCs can be used. In this work it is assumed that rail-mounted gantry (RMG) cranes (see Figure 2-4) [17] serve the AGVs in transfer points. Often two or more cranes are employed at one block to avoid operational interruptions in case of technical malfunctions and thereby increasing the productivity and reliability of the whole system. YCs can be man-driven, but in the highly automated seaports, there are autonomous gantry cranes in use, such as in Thamesport, Rotterdam and Hamburg [3].

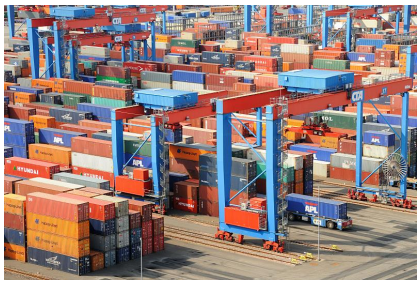

Figure 2-4: RMG cranes serve the AGVs in the yard.

2-3 Dimensions and performances

It is hard to present general numbers that define a container terminal, due to: fast developments in the container terminal equipment, the various different layouts that are possible and the wide spectrum of investments in harbors to stay competitive. It is however possible to give a good indication of how fast or accurate the equipment is in most (leading) container terminals. Data is used from various simulations of container terminals, such as the Delta Sealand (DSL) container terminal of Europe Container Terminals (ECT) Rotterdam at the Maasvlakte and the newest ECT terminal at the Maasvlakte 2 [18].

A vessel can carry up to 18.270 TEU (Mærsk Line Denmark). However, more frequently they will be around 13.000 TEU and with a typical length of approximately 350m. On deck the containers can be stowed 8 tiers high and 17 rows wide, in the hold 9 tiers high and 15 rows wide. These vessels are typically served by four to six QCs.

The performance of the QCs depends on the specifications and crane type. Simulations have been done with a cycle time of approximately 66 seconds (DSL terminal of ECT Rotterdam) [18], but also with a QC cycle time of 80 or 90 seconds [19].

In literature, the cycle time of a YC is often taken as two to four minutes, which are operational data collected from container terminals in Singapore and Hong Kong [20]. The exact time depends highly on where the container needs to be stacked or retrieved, since the yard can have large dimensions.

Every container terminal is unique, but a typical large scale terminal with perpendicular layout contains approximately:

- 10 QCs performing on average 30 moves per hour,
- 50 AGVs
- 20 to 30 YC modules, performing on average 10 to 15 moves per hour

and has a quay length of approximately 800m. The designed method should be able to provide a schedule for this problem size.

Chapter 3

Max-plus Algebra & Max-plus-linear Systems

Since max-plus-linear (MPL) systems are used in the approach to the AGV dispatching problem, it will first be explained why this approach is convenient. Hereafter, the related theory will be reviewed in this chapter. First the semantics of max-plus algebra are discussed. Hereafter, max-plus-linear-, switching max-plus-linear- and stochastic switching max-plus-linear systems are explained, expanding the complexity of the system in each step.

3-1 Max-plus algebra

The AGV dispatching problem can be modeled using discrete-event systems. These are event-driven dynamical systems, which means that their dynamics are due to asynchronous occurrences of discrete events. The state transitions are initiated by occurrences rather than a clock [21]. When max-plus algebra is applied to the scheduling of AGVs, it is possible to create a max-plus-linear (MPL) system, which is linear in the max-plus algebra. The class of MPL systems consists of discrete-event systems with synchronization but without concurrency or choise between rival transitions (Petri net theory) [22]. Synchronization requires the availability of several resources at the same time and all preceding operations need to be finished to start a new operation (concurrency) [23]. Using the max-plus algebra can be of great advantage, since [24]:

1. nonlinear systems can be presented as linear systems in the max-plus algebra,
2. it provides a structured way to determine bottlenecks in the system such as critical cranes and AGVs and it provides good initial scheduling variables. A system-theoretical approach based on eigenvalues and eigenvectors can be taken to determine the critical cycles,
3. there is a close relation between MPL models and a graph representation of a system, such that graph based methods can easily be used in the scheduling procedures,

4. when an AGV is late and delays become a problem, the MPL system can switch to a different mode, which changes the order of events to resolve delays. When an MPL system can switch modes, it is called switching max-plus-linear (SMPL) system. With max-plus algebra, the best switch can be calculated by using eigenvectors,
5. MPL models are often written implicit, but can easily be transformed into an explicit form, which *might* result in a faster computation time.

3-1-1 Semantics of max-plus algebra

Max-plus algebra uses mainly two operators, namely the maximization operator $\oplus$ and the addition operator $\otimes$. The definitions of these are

$$x \oplus y = \max(x, y) \qquad \text{and} \qquad x \otimes y = x + y \tag{3-1}$$

for $x, y \in \mathbb{R}_\varepsilon := \mathbb{R} \cup \{-\infty\}$ [25]. The symbols used for the maximization and addition operators resemble the conventional addition and multiplication symbols respectively. Therefore, the operator $\oplus$ is also named *max-plus-algebraic addition* and $\otimes$ the *max-plus-algebraic multiplication*. As in conventional algebra the max-plus-algebraic multiplication $\otimes$ has a higher priority than the max-plus-algebraic addition $\oplus$. An important difference with respect to the conventional algebra is that there is no inverse operation in the max-plus algebra.

The zero element for $\oplus$ is $\varepsilon := -\infty$. Using this definition, it holds that $\max(x, -\infty) = \max(-\infty, x) = x$, so that:

$$x \oplus \varepsilon = \varepsilon \oplus x = x \qquad \text{and} \qquad x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon.$$

The unit element is $e = 0$. All components of the max-plus algebra have been defined, which is the set $(\mathbb{R}_\varepsilon, \oplus, \otimes)$ [26]. The max-plus algebra is now denoted as:

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

It is also possible to take the power of $x \in \mathbb{R}$ by using $r \in \mathbb{R}$ and is denoted by $x^{\otimes r}$ [22]. This operation corresponds to rx in the conventional algebra, since

$$x^{\otimes r} = x \otimes x \otimes ... \otimes x = r \cdot x$$

for all $r \in \mathbb{Z}_+$. In Example (3-1) some numerical examples are listed.

Example 3-1.

$$\begin{array}{lll} 1 \otimes 3 \oplus \varepsilon & = \max(1+3, -\infty) & = 4 \\ e \otimes 2 \oplus 5 & = \max(0+2, 5) & = 5 \\ 2^{\otimes 3} & = 3 \times 2 & = 6 \end{array}$$

△

The algebraic properties of $\oplus$ and $\otimes$ are listed in Appendix A.

3-2 Matrix operations

The max-plus algebra can be extended to matrix operations [26]. The elements of a matrix $\mathbf{A}$ are denoted by:

$$a_{ij} = \Big[\mathbf{A}\Big]_{ij} \ , 1 \leq i \leq n \ , 1 \leq j \leq m.$$

The sum of the matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}_{\varepsilon}^{n \times m}$ is defined as

$$(\mathbf{A} \oplus \mathbf{B})_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \tag{3-2}$$

for all i, j. The matrix product of $\mathbf{A}$ and $\mathbf{C} \in \mathbb{R}_{\varepsilon}^{m \times p}$ is defined as

$$(\mathbf{A} \otimes \mathbf{C})_{ij} = \bigoplus_{k=1}^{m} a_{ik} \otimes c_{kj} = \max_{1 \leq k \leq m} (a_{ik} + c_{kj}) \tag{3-3}$$

for all i, j. In Eq. (3-3), $\mathbf{C}$ can be either a matrix ($p > 1$) or a vector($p = 1$). The scalar multiple of a matrix $\mathbf{A}$ is defined componentwise as

$$(c \otimes \mathbf{A})_{ij} = c \otimes a_{ij} = c + a_{ij} \tag{3-4}$$

for all i, j. Note the analogy with the conventional definitions of matrix sum and product. Lastly, there exists the max-plus dot product [26], which will not be used. An example of a matrix addition and matrix multiplication in the max-plus algebra is given in Example (3-2) (adapted from Duinkerken (2005) [27]).

Example 3-2. *Let*

$$\mathbf{A} = \begin{bmatrix} e & 1 \\ 1 & 2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & \varepsilon \\ 3 & 1 \end{bmatrix}.$$

The matrix addition of $\mathbf{A}$ *and* $\mathbf{B}$ *is:*

$$\mathbf{A} \oplus \mathbf{B} = \begin{bmatrix} e \oplus 1 & 1 \oplus \varepsilon \\ 1 \oplus 3 & 2 \oplus 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}.$$

The matrix product of $\mathbf{A}$ *and* $\mathbf{B}$ *becomes:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} (e \otimes 1) \oplus (1 \otimes 3) & (e \otimes \varepsilon) \oplus (1 \otimes 1) \\ (1 \otimes 1) \oplus (2 \otimes 3) & (1 \otimes \varepsilon) \oplus (2 \otimes 1) \end{bmatrix} = \begin{bmatrix} 1 \oplus 4 & \varepsilon \oplus 2 \\ 2 \oplus 5 & \varepsilon \oplus 3 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 5 & 3 \end{bmatrix}.$$

△

It should be noted that, opposite to the scalar max-plus algebra, the matrix multiplication fails to be commutative. Suppose the $\mathbf{A}$ and $\mathbf{B}$ matrices of Example (3-2). Indeed:

$$\mathbf{B} \otimes \mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \neq \begin{bmatrix} 4 & 2 \\ 5 & 3 \end{bmatrix} = \mathbf{A} \otimes \mathbf{B}.$$

Analogous to the zero element, there is a max-plus-algebraic zero matrix $(\mathcal{E}_{n\times m})_{ij} = \varepsilon$ for all i, j. Furthermore, the $n \times n$ identity matrix E_n in max-plus algebra is defined as $(\mathrm{E}_n)_{ii} = 0$ for all i and $(\mathrm{E}_n)_{ij} = \varepsilon$ for all i, j with $i \neq j$. Here it holds:

$$\mathbf{A} \oplus \mathcal{E}(n,m) = \mathcal{E}(n,m) \oplus \mathbf{A} = \mathbf{A} \quad \text{for } \mathbf{A} \in \mathbb{R}^{n\times m}$$

$$\mathbf{A} \otimes \mathrm{E}(m,m) = \mathrm{E}(n,n) \otimes \mathbf{A} = \mathbf{A} \quad \text{for } \mathbf{A} \in \mathbb{R}^{n\times m}.$$

3-3 Max-plus-linear systems

Discrete-event systems in which there is synchronization but no concurrency can be described by a model of the form:

$$\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1) \oplus \mathbf{B} \otimes \mathbf{u}(k) \tag{3-5}$$

$$\mathbf{y}(k) = \mathbf{C} \otimes \mathbf{x}(k) \tag{3-6}$$

with $\mathbf{A} \in \mathbb{R}_{\varepsilon}^{n\times n}$, $\mathbf{B} \in \mathbb{R}_{\varepsilon}^{n\times n_u}$ and $\mathbf{C} \in \mathbb{R}_{\varepsilon}^{n_y\times n}$ (n_u is the number of inputs, n_y the number of outputs), which are called max-plus-linear time-invariant discrete-event systems, or max-plus-linear (MPL) systems for short [23]. The components of the input, output and state are event times and k an event counter instead of a clock cycle. The AGV dispatching problem is scheduled using only the state and system matrix $\mathbf{A}$. In the state $\mathbf{x}(k)$ the starting times of events at k are stacked. In the $\mathbf{A}$-matrix the handling times are stored. The equations of the system can be derived using graph theory, as will become clear in the next section. For the theory to determine stability, controllability and observability it will be refered to Van den Boom et al. (2012) [28] and Gazarik et al. (1999) [29], but is not relevant to the AGV dispatching problem.

A simple MPL system for the container terminal is already designed by Contu et al. (2011) [30], describing only one QC, one YC and one AGV. Boetzelaer (2013) [31] expanded the framework to multiple cranes and AGVs with continues variables, modeling the system differently than will be shown in this work.

3-3-1 Relation to graph theory

In Section 3-1 it is stated that one of the great advantages of MPL systems is the relation to graph theory, which is consequently widely incorporated for system designing purposes. This is due to the definition of such an MPL system, that is closely related to a graph. The entries of the $\mathbf{A}$-matrix in the system equation as in Eq. (3-5) describe the graph and its weight on its edges. For example (adapted from Heidergott et al. (2005) [26]), when the $\mathbf{A}$-matrix is as follows:

$$\mathbf{A}_{\text{ex}} = \begin{bmatrix} \varepsilon & 15 & \varepsilon \\ \varepsilon & \varepsilon & 14 \\ 10 & \varepsilon & 12 \end{bmatrix},$$

the directed graph contains three nodes $\mathcal{V}(\mathbf{A}_{\text{ex}}) = \{1, 2, 3\}$, because $\mathbf{A}_{\text{ex}} \in \mathbb{R}^{3\times 3}$. For the elements that are ε, the directed graph does not have any connections, otherwise it has a

directed edge. In the case of the given $\mathbf{A}_{\text{ex}}$-matrix, there exist edges between (1,3), (2,1), (3,2) and (3,3), with weights 10, 15, 14 and 12 respectively. So the set of edges is $\mathcal{A}(\mathbf{A}_{\text{ex}}) = \{(1,3),(2,1),(3,2),(3,3)\}$. The graph representation can be seen in Figure 3-1. Note that for an arc (i,j) in graph $\mathcal{G}(\mathcal{V},\mathcal{A}) = \mathcal{G}(\mathbf{A}_{\text{ex}})$ the weight of arc (i,j) is given by a_{ji}. This is due to the max-plus equations arising from the system $\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1)$. For example, from the equation

$$x_1(k) = \big(\varepsilon \otimes x_1(k-1)\big) \oplus \big(15 \otimes x_2(k-1)\big) \oplus \big(\varepsilon \otimes x_3(k-1)\big)$$

it follows that there is a directed arc from $x_2(k-1)$ to $x_1(k)$.

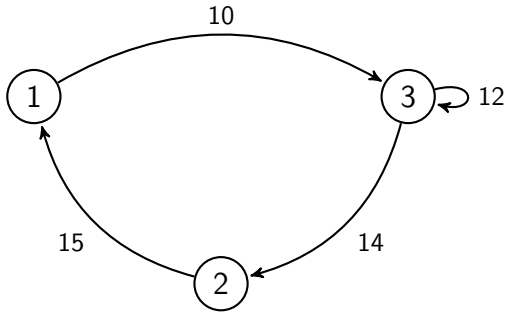

Figure 3-1: Directed graph representation of the matrix $\mathbf{A}_{\text{ex}}$.

The graph depicted in Figure 3-1 consists of two elementary circuits $p = ((1,3),(3,2),(2,1))$ and $q = (3,3)$, whereof the weights can be determined by:

$$\begin{aligned} |p|_w = &\quad a_{31} + a_{23} + a_{12} = 10 + 14 + 15 &&= 39 \\ |q|_w = &\quad\quad\quad a_{33} &&= 12. \end{aligned}$$

The length of a path is the number of nodes it contains ($|p|_\ell = 3$ and $|q|_\ell = 1$ in the example).

Critical circuits

Figure 3-1 consists of two *elementary* circuits. A circuit is a closed path and it is elementary when every node has only one incoming and one outgoing arc. If the weight is defined as

$$|\gamma|_w = \bigotimes_{k=0}^{\ell-1} a_{\eta_{h+k}\eta_{h+k+1}} = \mu^{\otimes \ell}, \tag{3-7}$$

then the **average weight** of a circuit can be determined by

$$\frac{|\gamma|_w}{|\gamma|_\ell} = \frac{1}{\ell} \times \mu^{\otimes \ell} = \mu. \tag{3-8}$$

The elementary path with the maximum average weight is called the **critical circuit**. Now let $\mathcal{C}(\mathbf{A})$ denote the set of elementary circuits in $\mathcal{G}(\mathbf{A})$, then the maximal average circuit weight and thus the critical circuit is:

$$\mu_c = \max_{p \in \mathcal{C}(\mathbf{A})} \frac{|p|_w}{|p|_\ell}. \tag{3-9}$$

Critical circuits are important to determine, since they are usually the bottlenecks of the scheduling problem and delays in these circuits might spread to the overall system. When the system is in operation, the critical circuits should always have the highest priority, since they are normative for the overall time that is needed to unload and reload a container ship (the makespan).

3-4 Switching max-plus-linear systems

The MPL system describes only **one mode**, or a single path in the system. In the container terminal, a mode corresponds to one assigned QC, one assigned AGV and one assigned YC. When there are more cranes and/or AGVs, more modes need to be introduced, creating a switching max-plus-linear (SMPL) system. The switching between modes changes the structure of the **A**-matrix, which is done with max-plus binary selection variables (Section 3-4-1). It also allows to break synchronization and change the order of events [25]. Every container that is transported by a different combination of QC, AGV and YC gets a unique $\mathbf{A}^{(\ell(k))}$-matrix describing the route and determining the time specifications of the container along the equipment. Eq. (3-5) is adapted to formulate the SMPL system for a container terminal:

$$\mathbf{x}(k) = \mathbf{A}^{(\ell(k))}(k) \otimes \mathbf{x}(k-1) \tag{3-10}$$

in which the matrix $\mathbf{A}^{(\ell(k))}$ is the system matrices in the ℓ-th mode. Each mode ℓ corresponds to a required set of synchronizations and an event order schedule. The switching between modes can be formulated in a switching rule (Appendix A). The fact that SMPL systems can handle a large amount of modes ℓ in limited time can also be seen in research to train scheduling [7, 32].

3-4-1 Max-plus binary control variables

The SMPL system can contain many different **A**-matrices, all describing a different path. As mentioned in Chapter 2, a typical dimension of a container terminal with a reasonable size is ten QCs, 50 AGVs and 20 YCs. This implies that for every isolated container within the system, there are $10 \times 50 \times 20 = 1000$ paths that need to be described in the model. To select the right path, binary control variables will be used. Instead of making 1000 different **A**-matrices, it is then possible to select the right entry of just one matrix. Suppose that it is possible to define straightforward the binary control variables for the QCs, YCs and AGVs, which are in the sets a, b and c respectively. Then the max-plus binary control variables would be:

$$\begin{aligned} a &= \{a_1, ..., a_{10}\} \\ b &= \{b_1, ..., b_{20}\} \\ c &= \{c_1, ..., c_{50}\}, \end{aligned}$$

where

$$a_i = \begin{cases} e & \text{if } i \text{ is selected} \\ \varepsilon & \text{if } i \text{ not selected} \end{cases}, \qquad b_j = \begin{cases} e & \text{if } j \text{ is selected} \\ \varepsilon & \text{if } j \text{ not selected} \end{cases},$$

$$c_h l = \begin{cases} e & \text{if } h \text{ is selected} \\ \varepsilon & \text{if } h \text{ not selected.} \end{cases}$$

The max-plus binary variables e and ε are equivalent to the plus-times binary variables 1 and 0 respectively. For an extensive example on max-plus binary control variables, it is referred to Appendix B.

3-5 Stochastic switching max-plus-linear systems

The SMPL system is deterministic when

- the switching is determined by a deterministic switching rule and
- all variables of the system are deterministic [33].

Accordingly, there can be two kind of uncertainties in the system. In the AGV dispatching problem the switching variable is not perturbed, such as in e.g. telecommunication networks [34]. In the container terminal system, the optimal schedule is determined, leaving no room for switching to a different mode to reconsider the order of events. Therefore, only the parametric uncertainty needs to be taken into account.

The parametric uncertainty is present in the system matrices, which are perturbed by noise (fast changes in the system matrices and variable disturbance) and modeling errors by assumption (slow changes or permanent and constant errors in the model variables). Two cases could be distinguished related to the characterization of the perturbation, which is either (1) bounded or (2) stochastic [21]. In the bounded case, it is assumed that it is not known how the uncertainty behaves and therefore the worst case scenario is assumed for every step. This is a safe choice in modeling the system. Due to the worst case assumption, it guarantees that all AGVs and cranes will be on time when no major external events happen such as the breakdown of equipment. This reduces the importance of on-line scheduling, but would adversely result in a slow scheduling scheme where QCs, AGVs and YCs have large waiting times. When stochastic perturbation is considered, it is possible to describe the behavior of the perturbation by a distribution function. A stochastic approach will lead to a tighter schedule than the bounded case. However, this also means that AGVs and cranes can be delayed and on-line scheduling will be important since it should resolve the delays.

The advantage of a tight schedule with minimal waiting time for the equipment is of such importance that the pros of the stochastic perturbation outweigh the pros of the bounded perturbation. Therefore, stochastic uncertainty is considered when the container jobs are modeled. A distribution of the perturbation becomes an assumption in the model of the

system and can be of wide variety. Ideally, the stochastic switching max-plus-linear (SSMPL) system is used to model the uncertainty, which is commonly additive for noise and model errors in plus-times linear systems. The plus operator in conventional algebra is equivalent to the max-plus multiplication operator, making the noise multiplicative in the SSMPL model [33, 35]. The resulting SSMPL system description:

$$\mathbf{x}(k) = \mathbf{A}^{(\ell(k))}(\mathrm{E}_u(k)) \otimes \mathbf{x}(k-1) \tag{3-11}$$

where $\mathbf{A}(\mathrm{E}_u(k))$ is the system matrix with the uncertainty vector $\mathrm{E}_u(k)$. An example is provided in Appendix C, where an MPL, SMPL and SSMPL model are derived for a production system (adapted from [25]).

Now that the theory on max-plus algebra and MPL systems is explained, the model of the container terminal can be designed.

Chapter 4

Model of the System

The main goal of exploring the possibilities and future perspectives for scheduling AGVs in an autonomous container terminal using MPL systems will be realized by optimizing a prediction model. The model is created by deriving the equations from a Petri net (PN)-like representation.

4-1 Model

The construction of the model can be divided into three steps, namely

- creating a graphical representation,
- deriving model equations,
- introducing binary control variables.

Each will be explained in the following sections.

4-1-1 Graphical representation

The model is constructed by using Petri net theory due to the relation between graph theory and MPL systems. A Petri net (PN) is a logical representation of the dynamics of a discrete-event system that presents the ordering of events in a plant [36]. A PN can either be timed or untimed. An untimed PN consists of a finite set of places P (circles), a finite set of transitions T (bars) and a set of directed arcs connecting the places and transitions. The PN represents the state of the system by the distribution of tokens in places. Usually, tokens are denoted by dots. A transition t is state-enabled when each upstream place contains at least one token, whereafter the transition can fire. When the transition is executed, an upstream token is removed and placed in (one of) the downstream place(s). In an untimed PN only the

order of events is specified, regardless of time aspects. However, in most applications time has to be considered, whereby a timed PN is obtained. The arrival times of the tokens in places are remembered in the state $x(k)$.

The system is modeled using three subcycles, namely the QC subcycle, the AGV subcycle and the YC subcycle. When two subcycles interact with each other, a synchronization occurs. In these subcycles, the following transitions occur:

- t_1: the AGV moves to the crane of the pick-up location (AGV cycle),
- t_4: the AGV moves to the crane of the drop-off location (AGV cycle),
- t_2: the QC moves to the ship to pick-up or drop-off a container (QC cycle),
- t_3: the QC (un)loads the AGV (QC/AGV synchronization),
- t_6: the YC moves to the yard to pick-up or drop-off a container (YC cycle),
- t_5: the YC (un)loads the AGV (YC/AGV synchronization).

Every transition t has a corresponding delay τ, which describes the time that a transition takes. One full cycle is the discharge or loading job of one container. Therefore, every container gets a new event step counter k. When k is a discharge job, the AGV gets loaded by a QC and unloaded by a YC. When k is a loading job, the AGV gets loaded by a YC and unloaded by a QC. In Figure 4-1 a general directed graph is given. Only one AGV, one QC and one YC is included. Expanding the graph with multiple AGVs and cranes is however easily done by adding arcs or nodes. The representation is not exactly equal to that of a PN, but the idea is equivalent. The arrays are transitions with a delay τ and the dots are places p. Whenever a place p is true, it receives a token. The filled dots need two places to be true and thus denote a synchronization. The open dots need only one token. The maximum time of two places to receive a token (causing a synchronization) is the time before a next transition is triggered.

To put the theory into perspective, suppose that only discharge jobs are considered. The process is explained using five nodes in Figure 4-1.

1. The QC picks up the container k from the vessel (1a) and moves it to the quay. When this is done, node 1 receives a token. Also the AGV needs be ready to receive the container and should move to the QC (1b). When this latter transition is complete, a token is placed at node 1. In the next transition (2a) the QC places the container on the AGV, which can only take place when both tokens are received.
2. The second node does not require a synchronization. It represents the moment that both the QC and AGV are ready to proceed. The QC will commence its next cycle and the AGV proceeds cycle k by moving the container to the yard.
3. Node 3 is again a synchronization node, following the same steps as node 1. The YC has just stored a container in the yard. Whenever it has moved back to the transfer

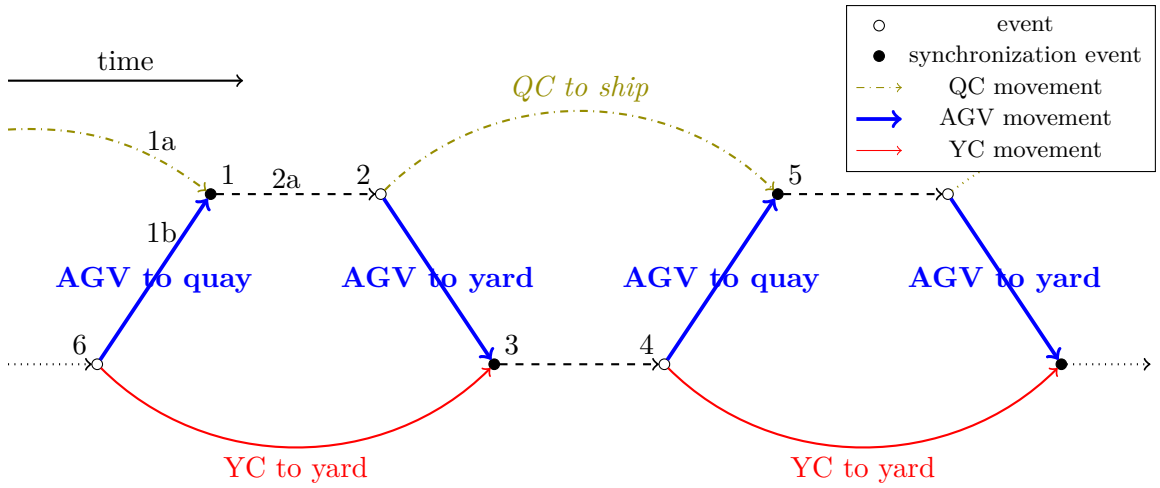

Figure 4-1: Graphical representation of the container terminal system.

point, node 3 receives a token. Node 3 receives also a token whenever the AGV has reached the yard. Only when it has received two tokens, the next transition can fire and the YC starts picking up the container from the AGV.

4. When the transition fires, node 3 is abandoned. Node 4 is reached with a certain delay, whereafter both the YC and AGV are ready to proceed. The YC will end cycle k by storing the container in the yard. The AGV will perform a next cycle and moves to the quay, reaching point 5.

5. Node 5 is comparable to node 1 and exists in a new cycle.

System analysis

In Section 3-3-1 the concept of critical circuits is explained. Using the handling times provided in Chapter 2 it is possible to determine the critical cycles in this system. First, the weights of the three cycles have to be determined. Suppose there is one QC, one AGV and one YC. The distance between the quay and yard equals 100m and the AGV average speed equals 2m/s. The following holds:

- weight of the QC cycle: $w_{QC} = 120$s (based on 30moves/h, including (un)loading time),
- weight of the YC cycle: $w_{YC} = 240$s (based on 15moves/h, including (un)loading time),
- weight of the AGV cycle: $w_{AGV} = \frac{200\text{m}}{2\text{m/s}} + (2 \times 15\text{s}) = 130$s (when time loss for (un)loading is 15s).

The graph in Figure 4-2 results from the statements about this simple system. The **A**-matrix that belongs to this configuration is

$$\mathbf{A} = \begin{bmatrix} 120 & 65 \\ 65 & 240 \end{bmatrix}$$

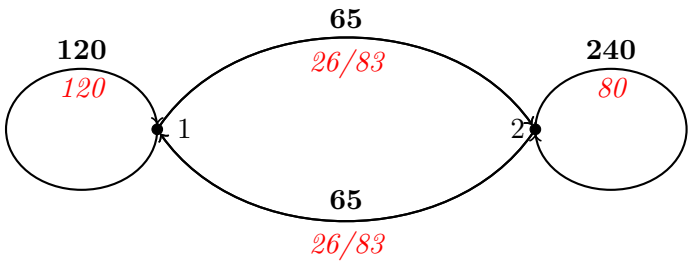

Figure 4-2: Graph of the container terminal system for a single mode (bold, black) and multiple modes (cursive, red). Node 1 and node 2 belong to the QC and YC synchronizations, respectively.

and the figure shows that {(1,1)}, {(1,2), (2,1)} and {(2,2)} are the circuits of the system. Now the critical cycle is the YC cycle, since $\max(120, 240, 130) = 240$. However, when *multiple modes* are considered, the ratio YC/QC typically is about two to three in a container terminal. Let us suppose there are three times more YCs than QCs. Furthermore, as an approximation, the number of AGVs is usually taken about five times the number of the QCs. However, presume that the travel distance might reach 800m, putting the weight on arcs (1,2) and (2,1) to $\left(\frac{800}{2} + 15\right) \times \frac{1}{5}$. These new weights for the multiple modes are displayed in red in Figure 4-2.

Interesting to point out is the shift of critical circuits. Now

$$w_{QC} = 120\text{s}, \qquad w_{YC} = 80\text{s}, \qquad w_{AGV} = 54 \text{ to } 166\text{s}.$$

Depending on which route the AGV takes, either the circuit of the QC or that of the AGV is critical. The boundaries on the system make it impossible to influence the time of the QC cycle. However, by assigning the AGVs correctly, the schedule can be minimized. This emphasizes the importance of solving the AGV dispatching problem.

4-1-2 Model equations

The model equations can directly be obtained from the graph in Figure 4-1. The general framework that is used as guidance for the definition of equation parameters has been formulated by Van den Boom et al. (2014) in [21]. It is designed for discrete event systems with semi-cyclic behavior. This type of behavior is specified by a periodically repeated set of operations. In semi-cyclic behavior the set of operations may vary over a limited set of possible sequences of operations. For now there is neither scheduling nor synchronization considered in the model design. A distinction can be made between four set of equations, since there are four cycle types:

1. only discharge jobs are considered ("ship-to-yard, ship-to-yard (SYSY)"),
2. only loading jobs are considered ("yard-to-ship, yard-to-ship (YSYS)"),
3. jumbled cycles: A discharge job is preceded by a loading job ("YSSY"),
4. jumbled cycles: A loading job is preceded by a discharge job ("SYYS").

The previous container jobs of the QC, YC and AGV in cycle k are independent of each other and therefore the cycle types of the different resources should be considered separately.

The equations can describe two possible discrete-time model forms: an implicit- or explicit model. When the model is explicit, the current cycle k only depends on the previous cycles. When it is implicit, the current cycle k depends on the previous cycles as well as on the current one. The implicit model requires less constraints and will therefore be an advantage in the MILP formulation. Since it will become clear that the number of constraints grow exponentially with the size of the problem, the implicit model is applied to keep the MILP proportions as small as possible. For a more extensive analysis on this statement, the interested reader is referred to Appendix D.

Deriving the discharge job equations

A job j is defined as the allocation of the j-th container from ship to yard in a discharge cycle. The event counter k is the allocation of the container from ship to yard at time event k. Every future job j for $j = 1, ..., N_p$ has $p_j = 6$ operations with the starting times in $\hat{x}_{p_j}(k+j)$. Job j has corresponding sequences of resources $\mathcal{R}_j = (r_{j,1}, ..., r_{j,p_j})$ and processing times in cycle k: $\mathcal{T}_j(k) = (\tau_{j,1}(k), ..., \tau_{j,p_j}(k))$. For container k, the following state is proposed:

$\hat{x}_1(k)$: starting time AGV travels to next QC location,
$\hat{x}_2(k)$: starting time QC retrieves container from vessel,
$\hat{x}_3(k)$: starting time QC loads AGV,
$\hat{x}_4(k)$: starting time transportation container by AGV from quay to yard,
$\hat{x}_5(k)$: starting time YC crane unloads AGV,
$\hat{x}_6(k)$: starting time YC stores container in the yard,

$\tau_1(k)$: time it takes AGV to travel to QC location,
$\tau_2(k)$: time it takes QC to retrieve container from vessel,
$\tau_3(k)$: time it takes QC to load AGV,
$\tau_4(k)$: time it takes AGV to transport container from quay to yard,
$\tau_5(k)$: time it takes YC to unload AGV,
$\tau_6(k)$: time it takes YC to store container in the yard.

The travel times are variable, whereas all QC and YC handling times are assumed constant. The equations are however still derived as if the handling times of the cranes can vary, since the SSMPL model will consider perturbed QC handling times. Moreover, in this fashion the YC handling times could easily be adjusted when the constant value would be substituted by a varying parameter depending on the container job location. An estimation of the AGV travel times can be made using its relation to the traveled distance and its speed, while accounting for some time loss due to delays.

When the states are assigned to the nodes, a new representation is obtained, as depicted in the directed graph of Figure 4-3.

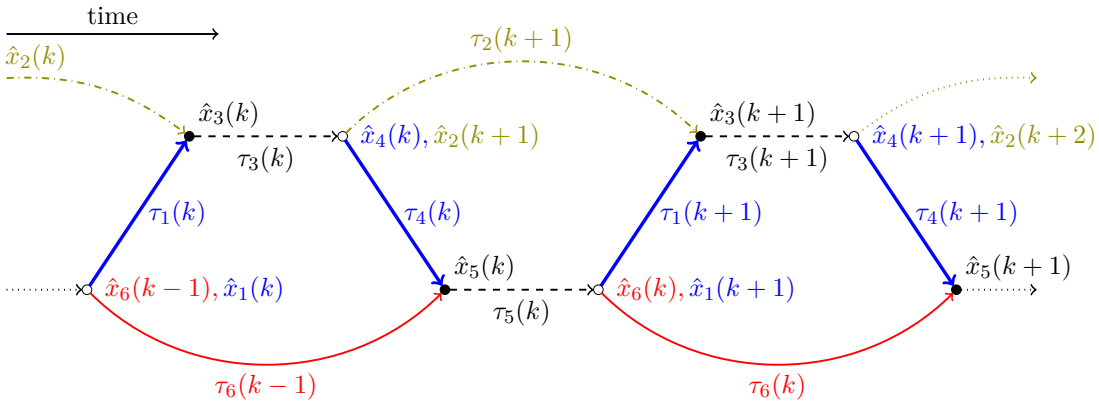

Figure 4-3: Graphical representation of the system cycles with corresponding states $\hat{x}$ and elapsed time τ.

The following matrix inequality can be derived using the graph in Figure 4-3:

$$\begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \\ \hat{x}_4(k) \\ \hat{x}_5(k) \\ \hat{x}_6(k) \end{bmatrix} \geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_1(k) & \tau_2(k) & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_3(k) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_4(k) & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_5(k) & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \\ \hat{x}_4(k) \\ \hat{x}_5(k) \\ \hat{x}_6(k) \end{bmatrix} \oplus \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_5^{(a)}(k-1) & \varepsilon \\ \varepsilon & \varepsilon & \tau_3^{(q)}(k-1) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_6^{(y)}(k-1) \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_1(k-1) \\ \hat{x}_2(k-1) \\ \hat{x}_3^{(q)}(k-1) \\ \hat{x}_4(k-1) \\ \hat{x}_5^{(a)}(k-1) \\ \hat{x}_6^{(y)}(k-1) \end{bmatrix} \tag{4-1}$$

where $(q), (a)$ and (y) denote the dependence of $(k-1)$ with the previous subcycles of the current QC, AGV and YC, respectively. It was already established that the synchronization points are in $\hat{x}_3$ and $\hat{x}_5$, where a maximum operator is applied. The states can be reduced by successive substitution of the equations in the two synchronization points. The previous cycle of the selected QC, AGV and YC are called respectively $(k-\mu_Q)$, $(k-\mu_A)$ and $(k-\mu_Y)$, were P the maximum of previous considered cycles: $\{\mu_Q,\ \mu_A,\ \mu_Y\} \leq P$. Now Eq. (4-1) can be used to obtain:

$$\begin{aligned} \hat{x}_1(k) &= \tau_5(k-\mu_A) \otimes x_5(k-\mu_A), \\ \hat{x}_2(k) &= \tau_3(k-\mu_Q) \otimes x_3(k-\mu_Q), \\ \hat{x}_4(k) &= \tau_3(k) \otimes x_3(k), \\ \hat{x}_6(k) &= \tau_5(k-\mu_A) \otimes x_5(k-\mu_A). \end{aligned}$$

The synchronization points can be described by the time that the QC can begin with loading the AGV:

$$\begin{aligned} \hat{x}_3(k) = \hat{x}_Q(k) \geq\ & \tau_2(k) \otimes \tau_3(k-\mu_Q) \otimes \hat{x}_Q(k-\mu_Q) \oplus \\ & \tau_1(k) \otimes \tau_5(k-\mu_A) \otimes \hat{x}_Y(k-\mu_A) \end{aligned} \tag{4-2}$$

and when the YC is able to start unloading the AGV:

$$\begin{aligned} \hat{x}_5(k) = \hat{x}_Y(k) \geq\ & \tau_3(k) \otimes \tau_4(k) \otimes \hat{x}_Q(k) \oplus \\ & \tau_5(k-\mu_Y) \otimes \tau_6(k-\mu_Y) \otimes \hat{x}_Y(k-\mu_Y). \end{aligned} \tag{4-3}$$

Notice that at the synchronization points, several equations can describe the same nodes, such as $\hat{x}_1(k)$ and $\hat{x}_6(k-\mu_A)$. It is possible to write Eq. (4-2) and Eq. (4-3) in matrix form as:

$$\begin{aligned}\begin{bmatrix}\hat{x}_Q(k)\\ \hat{x}_Y(k)\end{bmatrix} \geq &\left[\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline \tau_3(k)\otimes\tau_4(k) & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k)\\ \hat{x}_Y(k)\end{bmatrix} \oplus \\ &\left[\begin{array}{c|c} \tau_2(k)\otimes\tau_3(k-\mu_Q) & \varepsilon \\ \hline \varepsilon & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_Q)\\ \hat{x}_Y(k-\mu_Q)\end{bmatrix} \oplus \\ &\left[\begin{array}{c|c} \varepsilon & \tau_1(k)\otimes\tau_5(k-\mu_A) \\ \hline \varepsilon & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_A)\\ \hat{x}_Y(k-\mu_A)\end{bmatrix} \oplus \\ &\left[\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline \varepsilon & \tau_5(k-\mu_Y)\otimes\tau_6(k-\mu_Y) \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_Y)\\ \hat{x}_Y(k-\mu_Y)\end{bmatrix}\end{aligned} \tag{4-4}$$

$$\hat{\mathbf{x}}(k) \geq \mathbf{A}_0 \otimes \hat{\mathbf{x}}(k) \oplus \mathbf{A}_{\mu_Q} \otimes \hat{\mathbf{x}}(k-\mu_Q) \oplus \mathbf{A}_{\mu_A} \otimes \hat{\mathbf{x}}(k-\mu_A) \oplus \mathbf{A}_{\mu_Y} \otimes \hat{\mathbf{x}}(k-\mu_Y) \tag{4-5}$$

and is reformulated as

$$\hat{\mathbf{x}}(k) \geq \mathbf{A} \otimes \mathbf{x}(k). \tag{4-6}$$

The main issue with Eq. (4-5) is that the states consist of the synchronization events at just one QC and one YC. To address this issue, new states are defined containing the starting times of the synchronization of all QCs with an AGV to be loaded at event k and all YCs with an AGV to be unloaded at event k. The model will have the following state:

$$\begin{bmatrix}\hat{x}_1(k)\\ \vdots \\ \hat{x}_{QC_N}(k)\\ \hat{x}_{QC_{N+1}}(k)\\ \vdots \\ \hat{x}_{QC_N+YC_M}(k)\end{bmatrix} = \begin{bmatrix}\text{starting time last synchronization of } QC_1 \text{ with some AGV}\\ \vdots \\ \text{starting time last synchronization of } QC_N \text{ with some AGV}\\ \text{starting time last synchronization of } YC_1 \text{ with some AGV}\\ \vdots \\ \text{starting time last synchronization of } YC_M \text{ with some AGV}\end{bmatrix} \tag{4-7}$$

where the number of QCs are $n = 1, ..., N$ and the number of YCs are $m = 1, ..., M$.

When the state expands, the $\mathbf{A}$-matrix naturally expands, too. When $\hat{\mathbf{x}} \in \mathbb{R}^{(N+M)\times 1}$, then $\mathbf{A} \in \mathbb{R}^{(N+M)\times(N+M)}$ with the entries expanding in their own block matrix as in Eq. (4-4). In $\mathbf{A}_0$ there is a non-max-plus zero element in the lower left block matrix. This element describes the relation between the AGV travel time from *any* QC to *any* YC. Therefore, when expanding $\mathbf{A}_0$, all elements in $\mathbf{A}_0\{2,1\}$ equal $\tau_3(k) \oplus \tau_4(k)$. The same reasoning pattern can be used for the matrix $\mathbf{A}_{\mu_A}\{1,2\}$. On the contrary, $\mathbf{A}_{\mu_Q}\{1,1\}$ and $\mathbf{A}_{\mu_Y}\{2,2\}$ describe when the cranes *themselves* are ready for their next job. When expanding the state, the elements will only be on the diagonal of $\mathbf{A}_{\mu_Q}\{1,1\}$ and $\mathbf{A}_{\mu_Y}\{2,2\}$.

Deriving the loading job equations

The equations that have been derived are only applicable when subsequently discharge jobs are in order. It is also possible that loading jobs are performed, whereof the equations are

derived in a similar fashion. The difference is the definition of the states, model variables τ and the relations between them. They now get the following meaning:

$\hat{x}_1(k)$: starting time AGV travels to next YC location,
$\hat{x}_2(k)$: starting time QC stores container on vessel,
$\hat{x}_3(k)$: starting time QC unloads AGV,
$\hat{x}_4(k)$: starting time transportation container by AGV from yard to quay,
$\hat{x}_5(k)$: starting time YC crane loads AGV,
$\hat{x}_6(k)$: starting time YC retrieves container from yard,

$\tau_1(k)$: time it takes AGV to travel to YC location,
$\tau_2(k)$: time it takes QC to store container on vessel,
$\tau_3(k)$: time it takes QC to unload AGV,
$\tau_4(k)$: time it takes AGV to transport container from yard to quay,
$\tau_5(k)$: time it takes YC to load AGV,
$\tau_6(k)$: time it takes YC to retrieve container from the yard.

Using Figure 4-3 the equations can again be derived in a systematic way, following the same procedure as for the discharge moves, obtaining

$$\begin{bmatrix}\hat{x}_1(k)\\ \hat{x}_2(k)\\ \hat{x}_3(k)\\ \hat{x}_4(k)\\ \hat{x}_5(k)\\ \hat{x}_6(k)\end{bmatrix} \geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_3(k) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_4(k) & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_5(k) & \varepsilon \\ \tau_1(k) & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_6(k) \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix}\hat{x}_1(k)\\ \hat{x}_2(k)\\ \hat{x}_3(k)\\ \hat{x}_4(k)\\ \hat{x}_5(k)\\ \hat{x}_6(k)\end{bmatrix} \oplus \begin{bmatrix} \varepsilon & \varepsilon & \tau_3^{(a)}(k-1) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_2^{(q)}(k-1) & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_5^{(y)}(k-1) & \varepsilon \end{bmatrix} \otimes \begin{bmatrix}\hat{x}_1(k-1)\\ \hat{x}_2^{(q)}(k-1)\\ \hat{x}_3^{(a)}(k-1)\\ \hat{x}_4(k-1)\\ \hat{x}_5^{(y)}(k-1)\\ \hat{x}_6(k-1)\end{bmatrix}. \tag{4-8}$$

Finally, the minimal realization of the inequality (Eq. (4-9) and Eq. (4-10)) is retrieved. Again, the synchronization points are the time that the QC can start to unload the AGV:

$$\begin{aligned}\hat{x}_3(k) = \hat{x}_Q(k) \geq\ & \tau_2(k-\mu_Q) \otimes \tau_3(k-\mu_Q) \otimes \hat{x}_Q(k-\mu_Q) \oplus \\ & \tau_4(k) \otimes \tau_5(k) \otimes \hat{x}_Y(k)\end{aligned} \tag{4-9}$$

and the time when the YC can start to load the AGV:

$$\begin{aligned}\hat{x}_5(k) = \hat{x}_Y(k) \geq\ & \tau_1(k) \otimes \tau_3(k-\mu_A) \otimes \hat{x}_Q(k-\mu_A) \oplus \\ & \tau_6(k) \otimes \tau_5(k-\mu_Y) \otimes \hat{x}_Y(k-\mu_Y)\end{aligned} \tag{4-10}$$

or in matrix form:

$$\begin{bmatrix}\hat{x}_Q(k)\\ \hat{x}_Y(k)\end{bmatrix} \geq \left[\begin{array}{c|c} \varepsilon & \tau_4(k)\otimes\tau_5(k) \\ \hline \varepsilon & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k)\\ \hat{x}_Y(k)\end{bmatrix} \oplus \left[\begin{array}{c|c} \tau_2(k-\mu_Q)\otimes\tau_3(k-\mu_Q) & \varepsilon \\ \hline \varepsilon & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_Q)\\ \hat{x}_Y(k-\mu_Q)\end{bmatrix} \oplus \left[\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline \tau_1(k)\otimes\tau_3(k-\mu_A) & \varepsilon \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_A)\\ \hat{x}_Y(k-\mu_A)\end{bmatrix} \oplus \left[\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline \varepsilon & \tau_6(k)\otimes\tau_5(k-\mu_Y) \end{array}\right] \otimes \begin{bmatrix}\hat{x}_Q(k-\mu_Y)\\ \hat{x}_Y(k-\mu_Y)\end{bmatrix}. \tag{4-11}$$

Again the states and the **A**-matrix can be expanded when the system includes more than one mode.

Jumbled cycles

The QCs, YCs and AGVs can enter a discharge job or a loading job. However, when the discharge equations are inspected closely, it becomes clear that there is an indirect assumption that every discharge job is preceded by a discharge job. Considering the loading job equations, there is an indirect assumption that every loading job is preceded by a loading job. Therefore, a new distinction is made when discharging jobs are preceded by loading jobs, obtaining:

$$\begin{aligned}\hat{x}_Q(k) \geq\ & \tau_2(k)\otimes\tau_2(k-\mu_Q)\otimes\tau_3(k-\mu_Q)\otimes\hat{x}_Q(k-\mu_Q)\ \oplus \\ & \tau_1(k)\otimes\tau_3(k-\mu_A)\otimes\hat{x}_Q(k-\mu_A)\end{aligned} \tag{4-12}$$

$$\begin{aligned}\hat{x}_Y(k) \geq\ & \tau_3(k)\otimes\tau_4(k)\otimes\hat{x}_Q(k)\ \oplus \\ & \tau_5(k-\mu_Y)\otimes\hat{x}_Y(k-\mu_Y)\end{aligned} \tag{4-13}$$

and when loading jobs are preceded by discharging jobs, obtaining:

$$\begin{aligned}\hat{x}_Q(k) \geq\ & \tau_3(k-\mu_Q)\otimes\hat{x}_Q(k-\mu_Q)\ \oplus \\ & \tau_4(k)\otimes\tau_5(k)\otimes\hat{x}_Y(k)\end{aligned} \tag{4-14}$$

$$\begin{aligned}\hat{x}_Y(k) \geq\ & \tau_1(k)\otimes\tau_5(k-\mu_A)\otimes\hat{x}_Y(k-\mu_A)\ \oplus \\ & \tau_5(k-\mu_Y)\otimes\tau_6(k)\otimes\tau_6(k-\mu_Y)\otimes\hat{x}_Y(k-\mu_Y).\end{aligned} \tag{4-15}$$

The states are subsequently expanded again just as the **A**-matrix. Surely, not all entries of this expanded **A**-matrix should be finite, since not all cranes are handling a certain job k. The entries of those which are not selected should become minus infinite. This issue will be addressed by adopting max-plus binary control variables, to be discussed in Section 4-1-3.

One last equation needs to be added. The eight inequalities (defined in Eq. (4-2, 4-3, 4-9, 4-10, 4-12 – 4-15)) add handling times to the active states, while the nonactive states should not change. Therefore, the following inequality holds:

$$\hat{\mathbf{x}}(k) \geq \mathrm{E} \otimes \hat{\mathbf{x}}(k-1), \qquad \mathrm{E} \in \mathbb{R}^{(N+M)\times(N+M)} \tag{4-16}$$

where E the max-plus unity matrix. In this manner, the synchronization times of all cranes are "remembered", regardless if they are active in cycle k or not.

4-1-3 Binary control variables

Now that new cranes are added to the state, the serving cranes should be selected, since only these should receive a new ready time for their synchronization in job k. Max-plus binary control variables are already briefly discussed in Section 3-4-1. These variables are used to select the correct entries in the **A**-matrix. As became clear in Eq. (4-4) and Eq. (4-11), the **A**-matrix has four distinctive block matrices, existing for four cycles ($\mathbf{A}_0, \mathbf{A}_{\mu_Q}, \mathbf{A}_{\mu_Y}$ and $\mathbf{A}_{\mu_A}$). The block matrices are referred to as

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A_{qq}} & \mathbf{A_{qy}} \\ \hline \mathbf{A_{yq}} & \mathbf{A_{yy}} \end{array}\right]. \tag{4-17}$$

The max-plus binary control variables should be placed such that it is possible to uniquely select every path in the system. A distinction can be made between the max-plus binary control variables:

- category I – those which select the right elements in the block matrices of **A**,
- category II – those which make the proper part of an equation within an element of a block matrix the maximum,
- category III – those which select the right previous cycle.

Accordingly, the following max-plus binary control variables can be introduced:

Category I – variables which select the correct element in the block matrices

- $a(k)$: selects the serving QC in cycle k (in $\mathbf{A_{qq}}$, $\mathbf{A_{qy}}$),
- $b(k)$: selects the serving YC in cycle k (in $\mathbf{A_{yq}}$, $\mathbf{A_{yy}}$),
- $c(k)$: selects the previous crane of the AGV serving in cycle k (in **A**).

Category II – variables which maximize the correct part of the equation within an element in a block matrix

- $d_Q(k)$: selects the cycle type of the serving QC in cycle k (in $\mathbf{A_{qq}}$, $\mathbf{A_{yq}}$),
- $d_Y(k)$: selects the cycle type of the serving YC in cycle k (in $\mathbf{A_{qy}}$, $\mathbf{A_{yy}}$),
- $d_A(k)$: selects the cycle type of the serving AGV in cycle k (in **A**).

Category III – variables which select the correct previous cycle

- $m_Q(k)$: selects the previous cycle $(k - \mu_Q)$ of the serving QC in cycle k (in $\mathbf{A_{qq}}$),
- $m_Y(k)$: selects the previous cycle $(k - \mu_Y)$ of the serving YC in cycle k (in $\mathbf{A_{yy}}$),
- $m_A(k)$: selects the previous cycle $(k - \mu_A)$ of the serving AGV in cycle k (in $\mathbf{A}$).

Here, the cycle type can be either "SYSY"$^{(1)}$ (d_1), "SYYS"$^{(2)}$ (d_2), "YSYS"$^{(3)}$ (d_3) or "YSSY"$^{(4)}$ (d_4). These four cycles are equivalent to (1) discharge job followed by a discharge job, (2) discharge job followed by a loading job, (3) loading job followed by a loading job or (4) loading job followed by a discharge job. Since the previous cycles of the QC, YC and AGV are independent from each other, the cycle type of the QC (d_Q), YC (d_Y) and AGV (d_A) are also independent from each other. Moreover, it is stated that the m_* max-plus binary variables "select" the previous cycles. This regards the cycles in the following equation:

$$x(k) \geq \mathbf{A}_0 \otimes x(k) \oplus \mathbf{A}_1 \otimes x(k-1) \oplus \mathbf{A}_2 \otimes x(k-2) \oplus ... \oplus \mathbf{A}_P \otimes x(k-P) \tag{4-18}$$

where P the number of previous cycles that are considered.

All possible paths are described for $(k - P)$ up to k. By defining all these possibilities, only the right path needs to be selected with the max-plus binary control variables. Making this effort beforehand is convenient, since then the actual scheduling becomes less cumbersome. In the following example, the max-plus binary control variables are placed in the $\mathbf{A}$-matrices for a small case.

Example 4-3. *Suppose there are two QCs, three YCs and $P = 2$. The structure in the $\mathbf{A}$-matrix of the max-plus binary variables - category I will be*

$$\mathbf{A}_{p,I} = \left[\begin{array}{cc:ccc} a_1 \otimes c_1 & a_1 \otimes c_2 & a_1 \otimes c_3 & a_1 \otimes c_4 & a_1 \otimes c_5 \\ a_2 \otimes c_1 & a_2 \otimes c_2 & a_2 \otimes c_3 & a_2 \otimes c_4 & a_2 \otimes c_5 \\ b_1 \otimes c_1 & b_1 \otimes c_2 & b_1 \otimes c_3 & b_1 \otimes c_4 & b_1 \otimes c_5 \\ b_2 \otimes c_1 & b_2 \otimes c_2 & b_2 \otimes c_3 & b_2 \otimes c_4 & b_2 \otimes c_5 \\ b_3 \otimes c_1 & b_3 \otimes c_2 & b_3 \otimes c_3 & b_3 \otimes c_4 & b_3 \otimes c_5 \end{array}\right] \qquad \forall\ p \in P. \tag{4-19}$$

The structure of the second category that is defined, corresponds to the earlier derived equations with d_1 for Eq. (4-2)-(4-3), d_3 for Eq. (4-9)-(4-10), d_4 for Eq. (4-12)-(4-13) and d_2 for Eq. (4-14)-(4-15). As an example, let us consider $\mathbf{A_{qq}}(1,1)$ for any $p \in \{1, ..., P\}$:

$$\begin{aligned} \mathbf{A}_{\mathbf{qq},p,II}(1,1) = \ & \Big(\tau_1(k) \otimes \tau_3(k-p) \otimes d_{A_4}\Big) \oplus \Big(\tau_2(k) \otimes \tau_3(k-p) \otimes d_{Q_1}\Big) \oplus \ ... \\ & \Big(\tau_3(k-p) \otimes d_{Q_2}\Big) \oplus \Big(\tau_2(k-p) \otimes \tau_3(k-p) \otimes d_{Q_3}\Big) \oplus \ ... \\ & \Big(\tau_2(k) \otimes \tau_2(k-p) \otimes \tau_3(k-p) \otimes d_{Q_4}\Big) \ . \end{aligned} \tag{4-20}$$

The structure of category III will be

$$\mathbf{A}_{p,III} = \begin{bmatrix} m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} & \dots & m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} \\ m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} & \dots & m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} \\ m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} & \dots & m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} \\ m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} & \dots & m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} \\ m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} & \dots & m_{Q,p} \oplus m_{Y,p} \oplus m_{A,p} \end{bmatrix} \qquad p = 1, 2. \quad (4\text{-}21)$$

Note the differences in using multiplication $\otimes$ and addition $\oplus$ in the first and third matrix. When $\otimes$ is used, an element is only selected when all control variables select. When $\oplus$ is used, only a single control variable needs to select to have a non-zero element in the matrix. $\triangle$

When a binary control variable activates an element of the desired path through the system, it equals the unit element e. It then adds zero to the handling times (e.g. consider the matrices of Example (4-3)). When the binary control variable inactivates, it equals the zero element $-\infty$, such that the element will never be a maximum. To use this latter control variable other than in the notations is unwise. Computations using ∞ can result in an indeterminate form (such as $\infty - \infty$ or $0 \times \infty$) which are undefined. Instead, the parameter β is used to resemble $-\infty$, which is defined as:

$$\beta << 0. \quad (4\text{-}22)$$

Predefined and scheduling variables

A last observation on the max-plus binary control variables is that a distinction can be made between predefined and scheduling control variables. The predefined ones are known in the optimization, while the scheduling max-plus binary control variables are unknown beforehand. These latter are used to assign jobs to the AGVs. The work queue determines which QC, denoted by $a(k)$, and which YC, denoted by $b(k)$, should be selected in the particular container job k. It indirectly determines which previous QC cycle $m_Q(k)$ and previous YC cycle $m_Y(k)$ are selected. Furthermore, the information on the cycle type for the QC and YC ($d_Q(k)$ and $d_Y(k)$) in cycle k is available by using the work queue in combination with $m_Q(k)$ and $m_Y(k)$.

Unknown max-plus binary control variables are $c(k)$, $m_A(k)$ and $d_A(k)$. Both the previous crane $c(k)$ of some AGV and the previous cycle $m_A(k)$ basically schedule the AGVs. Subsequently, the cycle type of the AGV $d_A(k)$ can be calculated.

4-2 Model reduction

Writing all the inequality constraints down that the SMPL equations directly impose, results in an enormous amount of inequality constraints and variables. Since the aim is to be able to determine a schedule on-line, the computation speed matters. It is therefore wise to reduce the model as much as possible, since otherwise two main problems can be identified.

1. Creating the matrices can take a long time (matter of seconds to minutes for large cases[1]). Fortunately, most of the MILP formulation is generated only once and can be reused.
2. More importantly, the solution time of the optimization depends on the number of constraints and variables. When the optimization need to satisfy less constraints or variables, it is likely that the solution time is faster.

By taking a closer look at the equations, it can be seen that the first part of Eq. (4-3) equals the first part of Eq. (4-13) and the second part of Eq. (4-9) equals the second part of Eq. (4-14). Therefore, these parts of Eq. (4-13) and Eq. (4-14) can be removed. The model can be reduced even more by assuming that τ_2 and τ_6 remain (possibly) variable and that:

- the time a QC takes to (un)load an AGV is the same in every cycle k: $\tau_3(k) = \tau_3$,
- the time a YC takes to (un)load an AGV is the same in every cycle k: $\tau_5(k) = \tau_5$,

and knowing that:

- the travel times to a begin location ($\tau_1(k)$) and between cranes ($\tau_4(k)$) **never** appear in the same constraint. One general τ_v replaces the two travel times,
- the movements of the AGVs are not mapped directly.

The first three points reduce the number of handling time parameters (τ-s) in the system to suppress the size of the MILP vectors to be introduced in Chapter 5. The last provided point is less obvious. It would be possible to map the movement of every AGV such that it is stated within the constraints that an AGV performed in cycle k. Every constraint that contains a travel time τ_1 or τ_4 would then be repeated V times, if V would be the total number of AGVs in the system. If the AGVs are unspecified, the constraints do not have to be repeated (as was already implied when defining the binary control variables). It is not necessary to use different travel time indications for every vehicle ($\tau_{v,1}, ..., \tau_{v,V}$). It is then however not possible to track the movements of the AGVs directly, but these can still be calculated by using the previous served cycle of the AGV m_A and the work queue.

Initially, the number of constraints and handling times equaled:

$$nr_{constr} = (4NP + 4MP + N^2VP + M^2VP + 4NMV + 4NMVP + N + M) \times N_p$$
$$nr_{hand} = (V + N + N + V + M + M) \times (N_p + V),$$

where N_p the number of jobs that are to be scheduled: the prediction horizon. For a medium sized container terminal of five QCs, fifteen YCs, 25 AGVs and 40 previously considered cycles, this would result in 8.410.800 constraints. When the AGVs are not mapped, the number of constraints equal

$$nr_{constr} = (4NP + 4MP + N^2P + M^2P + 4NM + 4NMP + N + M) \times N_p$$
$$nr_{hand} = (1 + N + 1 + 1 + 1 + M) \times (N_p + V).$$

[1] computer specifications in Section 5-3

Only 382.800 constraints are left of the medium sized container terminal case. When double inequality constraints are removed, the number of constraints equal

$$nr_{constr} = (4NP + 4MP + N^2P + M^2P + 2NM + 4NMP + N + M) \times N_p$$

and only 380.550 constraints remain for the example case. This latter reduction in constraints introduces two new optimization variables due to the selection of cycle types:

$$nr_{var} = (N + M + 4 + P)) \times N_p \quad \rightarrow \quad (N + M + 6 + P)) \times N_p.$$

The exact formulation and meaning of these two new variables is to be discussed in Section 5-1-1, Eq. (5-8) and 5-9.

Sparse matrices

The use of sparse matrices reduces the memory capacity that is necessary. In sparse matrices, only matrix elements unequal to zero are stored, while all matrix elements that are equal to zero are disregarded. The effect on the memory usage is shown in Table E-1 and Table E-2 in Appendix E.

Chapter 5

MILP & Benders' Decomposition

Now that the model is defined, there are numerous ways to solve the problem. One convenient method is recasting the problem in a MILP formulation and using a (MILP) solver to obtain an optimum. Another approach to retrieve an optimal schedule is through Benders' decomposition, which is often used in stochastic programming. First, the MILP is generated, wereafter the Benders' decomposition is explained. The third part of this chapter shows the results of both the optimization approaches.

5-1 MILP framework

Since the model contains real and integer parameters to optimize, the problem is defined as a mixed-integer programming (MIP) problem. Moreover, since the problem is formulated in the max-plus algebra, it is also linear. Therefore, a mixed-integer linear programming (MILP) problem formulation can be used with integers that are either known variables (w) or to-be scheduled variables (z). When β (defined in Eq. (4-22)) is placed before the matrix-vector multiplication, binary variables can be used. Instead now "select" corresponds to zero and "do not select" corresponds to one. This results in the MILP form:

$$\min_{\mathbf{x},\mathbf{z}} J(\mathbf{x},\mathbf{z})$$

subjected to

$$\begin{aligned} \mathbf{E}\mathbf{x} + \beta\mathbf{F}\mathbf{z} &\leq \mathbf{B}\boldsymbol{\Theta} + \beta\mathbf{K}\mathbf{w} \\ \mathbf{E}_{\mathbf{eq}}\mathbf{x} + \beta\mathbf{F}_{\mathbf{eq}}\mathbf{z} &= \mathbf{B}_{\mathbf{eq}}\boldsymbol{\Theta} + \beta\mathbf{K}_{\mathbf{eq}}\mathbf{w} \\ \mathbf{x} \in \mathbb{R}^{n_r},\ \mathbf{z} &\in \mathbb{Z}^{n_i}. \end{aligned} \tag{5-1}$$

Here, the parameters $\mathbf{x}$, $\mathbf{z}$ and Θ are:

$$\mathbf{x}(k) = \begin{bmatrix} x(k+N_p-1) \\ \vdots \\ x(k+1) \\ x(k) \\ x(k-1) \\ \vdots \\ x(k-V) \end{bmatrix}, \qquad \mathbf{\Theta}(k) = \begin{bmatrix} \theta(k+N_p-1) \\ \vdots \\ \theta(k+1) \\ \theta(k) \\ \theta(k-1) \\ \vdots \\ \theta(k-V) \end{bmatrix}, \qquad \mathbf{z}(k) = \begin{bmatrix} z(k+N_p-1) \\ \vdots \\ z(k+1) \\ z(k) \end{bmatrix}$$

where

$$x(k) = \begin{bmatrix} \hat{x}_{QC_1}(k) \\ \vdots \\ \hat{x}_{QC_N}(k) \\ \hat{x}_{YC_1}(k) \\ \vdots \\ \hat{x}_{YC_M}(k) \end{bmatrix}, \qquad \theta(k) = \begin{bmatrix} \tau_v(k) \\ \tau_{2,1}(k) \\ \vdots \\ \tau_{2,n}(k) \\ \tau_3 \\ \tau_5 \\ \tau_{6,1}(k) \\ \vdots \\ \tau_{6,m}(k) \end{bmatrix}, \qquad z(k) = \begin{bmatrix} c_1(k) \\ \vdots \\ c_{n+m}(k) \\ m_{A_1}(k) \\ \vdots \\ m_{A_p}(k) \\ d_{A_1}(k) \\ \vdots \\ d_{A_6}(k) \end{bmatrix}.$$

Note that, even though P ($P \geq V$) previous cycles are considered, the state and time vector only acknowledge $k - V$ cycles. Including the constraints proposed later in this section, this allows V number of paths through the system, hence there are V AGVs.

In the vectors described, $\mathbf{x}$ is the state, $\mathbf{z}$ are all the unknown and to-be-scheduled max-plus binary control variables and $\mathbf{\Theta}$ the time vectors. Furthermore, the parameter $\mathbf{w}$:

$$\mathbf{w}(k) = \begin{bmatrix} w(k+N_p-1) & \dots & w(k+1) & w(k) & w(k-1) & \dots & w(k-V) \end{bmatrix}^T,$$

with $w(k)$:

$$w(k) = \Big[a_1(k) \quad \dots \quad a_n(k) \quad b_1(k) \quad \dots \quad b_m(k) \quad m_{Q_1}(k) \quad \dots \quad m_{Q_P}(k) \quad \dots \\ m_{Y_1}(k) \quad \dots \quad m_{Y_P}(k) \quad d_{Q_1}(k) \quad \dots \quad d_{Q_4}(k) \quad d_{Y_1}(k) \quad \dots \quad d_{Y_4}(k) \Big]^T$$

describes all the known max-plus binary control variables for every container job. If the above definitions of the variables are examined closely, it can be seen that there exist six d_A in the unknown vector $z(k)$. The fifth and sixth d_A are as follows:

$$d_{A_5} = d_{A_1} \oplus d_{A_4}$$
$$d_{A_6} = d_{A_2} \oplus d_{A_3}$$

and became a necessity due to the merging of the equations to reduce the model size.

To easily obtain the matrices $\mathbf{E}, \mathbf{F}, \mathbf{K}$ and $\mathbf{B}$ for the MILP framework, the $\mathbf{A}$-matrix is parameterized (Example (5-4)). In the parameterization the following is defined: the job number (c_j), the position in the $\mathbf{A}$-matrix (i, j), the previous cyclus (μ), the index of all τ, stored in three columns $\begin{bmatrix}\theta^1 & \theta^2 & \theta^3\end{bmatrix}$, the index of all known binary selection variables, stored in $\begin{bmatrix}w^1 & w^2 & w^3\end{bmatrix}$ and the index of all unknown binary selection variables in $\begin{bmatrix}z^1 & z^2 & z^3\end{bmatrix}$.

Example 5-4. *Let us consider the model equation Eq. (4-2):*

$$\begin{aligned}\hat{x}_Q(k) \geq \tau_2(k) \otimes \tau_3 \otimes \hat{x}_Q(k-\mu_Q) \oplus \\ \tau_1(k) \otimes \tau_5 \otimes \hat{x}_Y(k-\mu_A).\end{aligned}$$

This equation satisfies the two inequalities:

$$\begin{aligned}\hat{x}_Q(k) \geq \tau_2(k) \otimes \tau_3 \otimes \hat{x}_Q(k-\mu_Q), \\ \hat{x}_Q(k) \geq \tau_1(k) \otimes \tau_5 \otimes \hat{x}_Y(k-\mu_A).\end{aligned}$$

Suppose there is one QC with state (x_1) and two YCs (x_2 and x_3), $P = 2$ and only one container job j is considered ($N_p = 1$). Following from Eq. (5-1):

$$\mathbf{x}(k) = \begin{bmatrix}\hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \\ \hat{x}_1(k-1) \\ \hat{x}_2(k-1) \\ \hat{x}_3(k-1) \\ \hat{x}_1(k-2) \\ \hat{x}_2(k-2) \\ \hat{x}_3(k-2)\end{bmatrix}, \quad \mathbf{\Theta}(k) = \begin{bmatrix}\tau_v(k) \\ \tau_2(k) \\ \tau_3 \\ \tau_5 \\ \tau_{6,1}(k) \\ \tau_{6,2}(k)\end{bmatrix}, \quad \mathbf{z}(k) = \begin{bmatrix}c_1(k) \\ c_2(k) \\ c_3(k) \\ m_{A_1}(k) \\ m_{A_2}(k) \\ d_{A_1}(k) \\ d_{A_2}(k) \\ d_{A_3}(k) \\ d_{A_4}(k)\end{bmatrix} \tag{5-2}$$

and

$$\begin{aligned}\mathbf{w}(k) = \Big[a_1(k) \quad b_1(k) \quad b_2(k) \quad m_{Q_1}(k) \quad m_{Q_2}(k) \quad m_{Y_1}(k) \quad m_{Y_2}(k) \quad \dots \\ d_{Q_1}(k) \quad d_{Q_2}(k) \quad d_{Q_3}(k) \quad d_{Q_4}(k) \quad d_{Y_1}(k) \quad d_{Y_2}(k) \quad d_{Y_3}(k) \quad d_{Y_4}(k)\Big]^T.\end{aligned} \tag{5-3}$$

Remembering that max-plus binary parameter "a" selects the QC, "b" the YC, "m_Q" the previous QC cycle, "m_Y" the previous YC cycle, "d_Q" the QC cycle type and "d_Y" the YC cycle type, the parameterization for the inequality following from Eq. (4-2) is shown in Table 5-1. All $c_j = 1$ for every row in the table. The second inequality that follows from Eq. (4-2) introduces more constraints and is shown in Table 5-2.

i	j	p	θ^1	θ^2	θ^3	w^1	w^2	w^3	z^1	z^2	z^3
1	1	1	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_1}^p(k)=4$	$d_{Q_1}^p(k)=8$	0	0	0
1	1	2	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_2}^p(k)=5$	$d_{Q_1}^p(k)=8$	0	0	0
1	1	1	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_1}^p(k)=4$	$d_{Q_2}^p(k)=9$	0	0	0
1	1	2	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_2}^p(k)=5$	$d_{Q_2}^p(k)=9$	0	0	0
1	1	1	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_1}^p(k)=4$	$d_{Q_3}^p(k)=10$	0	0	0
1	1	2	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_2}^p(k)=5$	$d_{Q_3}^p(k)=10$	0	0	0
1	1	1	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_1}^p(k)=4$	$d_{Q_4}^p(k)=11$	0	0	0
1	1	2	$\tau_2^p(k)=2$	$\tau_3^p=3$	0	$a_1^p(k)=1$	$m_{Q_2}^p(k)=5$	$d_{Q_4}^p(k)=11$	0	0	0

Table 5-1: Parameterization of the first part of Eq. (4-2).

c_j	i	j	p	θ^1	θ^2	θ^3	w^1	w^2	w^3	z^1	z^2	z^3
1	1	2	1	1	4	0	1	0	0	2	4	6
1	1	2	2	1	4	0	1	0	0	2	5	6
1	1	3	1	1	4	0	1	0	0	3	4	6
1	1	3	2	1	4	0	1	0	0	3	5	6
1	1	2	1	1	4	0	1	0	0	2	4	7
1	1	2	2	1	4	0	1	0	0	2	5	7
1	1	3	1	1	4	0	1	0	0	3	4	7
1	1	3	2	1	4	0	1	0	0	3	5	7
1	1	2	1	1	4	0	1	0	0	2	4	8
1	1	2	2	1	4	0	1	0	0	2	5	8
1	1	3	1	1	4	0	1	0	0	3	4	8
1	1	3	2	1	4	0	1	0	0	3	5	8
1	1	2	1	1	4	0	1	0	0	2	4	9
1	1	2	2	1	4	0	1	0	0	2	5	9
1	1	3	1	1	4	0	1	0	0	3	4	9
1	1	3	2	1	4	0	1	0	0	3	5	9

Table 5-2: Parameterization of the second part of Eq. (4-2).

△

The parameterization of the **A**-matrix is useful, since it prescribes most of the constraints in the MILP problem. Its columns can be created when the number of QCs and YCs, P and N_p are known. In Example (5-4) it is clear that the values in several columns are pointers to the vectors of $\mathbf{x}$, $\mathbf{\Theta}$, $\mathbf{w}$ and $\mathbf{z}$. This way, it is not obligated to define any specific information on the model parameters and therefore it serves as a general framework that only needs N, M, P and N_p.

If all the model equations are parameterized, the number of constraints equals:

$$nr_{constr} = \sum_{j=0}^{N_p-1} \Big(4 \times N \times \min(P, V+j) + 2 \times N \times M + (N+M)^2 \times \min(P, V+j) \; ... \\ + \; 4 \times M \times \min(p, V+j) + N + M\Big). \quad (5\text{-}4)$$

The constraints describe every allowed path in the system for every container job j. This includes all the resources (QCs, YCs) with AGVs traveling all paths, which is illustrated in Figure 5-1. The figure reinforces what Eq. (5-4) also shows: the number of defined paths (and thus constraints) grow rapidly when the container terminal size increases.

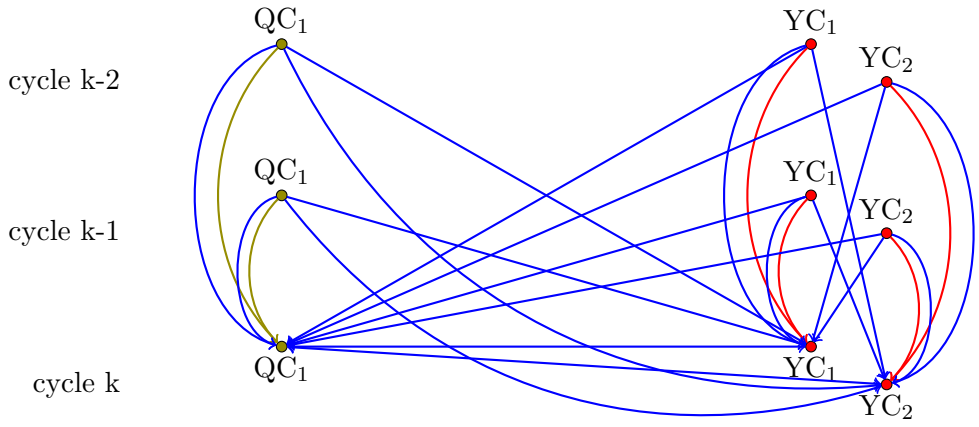

Figure 5-1: All paths described by the **A**-matrix parameterization for one container job j when there are one QC, two YCs and $P = 2$. The paths of the QC and YCs are green and red, respectively. The AGV paths occupy the largest number of directed arcs, which are drawn in blue.

Beside the constraints that arise from the parameterization of the **A**-matrix, several other important constraints are necessary to define. These will set boundary conditions which the optimization should satisfy.

5-1-1 Constraints

In the previous section, most inequality constraints are already obtained by recasting the model into the MILP framework of Eq. (5-1), describing every possible travel path in the system. The equality constraints that determine the restrictions and relations between the binary variables are introduced next (remember that zero corresponds to "select" and one to "not select").

1. Per cycle k, only one previous crane c can be selected

$$c_1(k+j) \otimes ... \otimes c_{N+M}(k+j) = N + M - 1 \qquad \forall j. \tag{5-5}$$

2. Per cycle k, only one previous cycle m_A can be selected

$$m_{A_1}(k+j) \otimes ... \otimes m_{A_P}(k+j) = P - 1 \qquad \forall j. \tag{5-6}$$

3. Per cycle k, only one cycle type of d_{A_1} to d_{A_4} can be selected

$$d_{A_1}(k+j) \otimes ... \otimes d_{A_4}(k+j) = 3 \qquad \forall j. \tag{5-7}$$

4. Whenever d_{A_1} or d_{A_4} is selected, d_{A_5} is also selected (arising from model reduction)

$$-d_{A_1}(k+j) \otimes -d_{A_4}(k+j) \otimes d_{A_5}(k+j) = -1 \qquad \forall j. \tag{5-8}$$

5. Whenever d_{A_2} or d_{A_3} is selected, d_{A_6} is also selected (arising from model reduction)

$$-d_{A_2}(k+j) \otimes -d_{A_3}(k+j) \otimes d_{A_6}(k+j) = -1 \qquad \forall j. \tag{5-9}$$

6. Per cycle k, there is only one outgoing arc from a node in the previous cycles to cycle k

$$\begin{aligned} &m_{A_1}(k+j-1) \otimes m_{A_2}(k+j-2) \otimes ... \\ &\quad \otimes m_{A_P}(k+j-\min(V+j,P)) = \min(V+j,P)-1 \qquad \forall j. \end{aligned} \tag{5-10}$$

7. Last V cycles in $\mathbf{x}(k)$ equal the initial state $\mathbf{x}_{init}$

$$\begin{bmatrix} x(k-1) & \dots & x(k-V) \end{bmatrix}^T = \mathbf{x}_{init}.$$

8. First entry in $\mathbf{z}(k)$ equals zero, a dummy variable due to the parameterization

$$z_1 = 0 \qquad \forall j.$$

9. Every cycle can be a termination cycle for an AGV and can result in a sink node

$$\mathbf{s} = \begin{bmatrix} s_1 & \dots & s_{N_p+V} \end{bmatrix}^T = nr_{nodes} - nr_{AGVs} = N_p.$$

To match the binary control variables to the correct cycle, inequality constraints are introduced:

1. A particular crane c belongs to previous cycle $k+j-p$

$$\begin{array}{rcccccrcc} -m_{A_1}(k+j) \otimes -c_1(k+j) & \leq & \nu_1^c & \dots & -m_{A_P}(k+j) \otimes -c_1(k+j) & \leq & \nu_1^c \\ & \vdots & & & & \vdots & \\ -m_{A_1}(k+j) \otimes -c_{N+M}(k+j) & \leq & \nu_{N+M}^c & \dots & -m_{A_P}(k+j) \otimes -c_{N+M}(k+j) & \leq & \nu_{N+M}^c \end{array} \tag{5-11}$$

where

$$\nu^c = \begin{cases} 0 & \text{if crane } \nu^c \text{ served in cycle } k+j-p \\ -1 & \text{if crane } \nu^c \text{ did not serve in cycle } k+j-p \end{cases} \qquad \forall p,\ \forall j.$$

2. A particular cycle type d_A belongs to previous cycle $k+j-p$ (inequality constraint)

$$\begin{array}{rcccccrcc} -m_{A_1}(k+j) \otimes -d_{A_1}(k+j) & \leq & \nu_1^d & \dots & -m_{A_p}(k+j) \otimes -d_{A_1}(k+j) & \leq & \nu_1^d \\ & \vdots & & & & \vdots & \\ -m_{A_1}(k+j) \otimes -d_{A_4}(k+j) & \leq & \nu_4^d & \dots & -m_{A_p}(k+j) \otimes -d_{A_4}(k+j) & \leq & \nu_4^d \end{array} \tag{5-12}$$

where

$$\nu^d = \begin{cases} 0 & \text{if cycle of AGV } \nu^d \text{ corresponds to } d_A \\ -1 & \text{if cycle of AGV } \nu^d \text{ does not correspond to } d_A \end{cases} \qquad \forall p,\ \forall j.$$

5-1-2 Performance index

The performance index describes the cost of the schedule, which should be minimized. How this is calculated exactly, depends on the objective of the optimization. An obvious objective would be to minimize the makespan (the last synchronization of the last performing QC). It is e.g. also possible to minimize the sum of the last QC times or the AGV travel distance (an "ecological" approach regarding fuel consumption). For the model that is presented so far, several performance indices will be evaluated.

- The last synchronization times of the QCs. Several possibilities are reviewed, namely (I) minimizing the sum of these times, (II) minimizing the maximum of these times (makespan) and (III) making trade-offs between the two possible objectives. The performance index that is obtained:

$$\min_{x,z,t_n} J = \min_{x,z,t_n} \left(\lambda_{sum} \sum_{n=1}^{N} \hat{x}_n(k+N_p) + \lambda_{max} \max_{n\in(1,\dots,N)} \hat{x}_n(k+N_p) \right). \tag{5-13}$$

  (I) To minimize the sum of the last QC synchronization times, $\lambda_{max} = 0$ and the first N elements in $\mathbf{x}(k)$ are weighted by $\lambda_{sum} = 1$.
  (II) To minimize the maximum of the last QC synchronization times, $\lambda_{max} = 1$ and $\lambda_{sum} = 0$. To obtain the maximum time, a new variable t_n is introduced, where

$$t_n \geq \hat{x}_1(k+N_p) \oplus \dots \oplus \hat{x}_N(k+N_p) \tag{5-14}$$

  Accordingly, the inequalities for the MILP formulation become:

$$\begin{aligned} t_n &\geq \hat{x}_1(k+N_p) \\ &\vdots \\ t_n &\geq \hat{x}_N(k+N_p). \end{aligned}$$

  (III) The trade-off between the average performance of the QCs (sum) and the worst performance (max) is done by setting $\lambda_{sum} = 1$ and $\lambda_{max} = \begin{bmatrix} 0.25 & 0.50 & 0.75 & 1 \end{bmatrix}$.

- The last synchronization times of the YCs. Again, different possibilities are analyzed: minimizing the sum and the maximum of the YC times and finally making trade-offs between the two possible objectives. Exactly the same weights are used as in the minimizations of the QC performances, except now the minimization is done over the YC synchronization times:

$$\min_{x,z,t_m} J = \min_{x,z,t_m} \left(\lambda_{sum} \sum_{m=1}^{M} \hat{x}_{N+m}(k+N_p) + \lambda_{max} \max_{m\in(1,\dots,M)} \hat{x}_{N+m}(k+N_p) \right). \tag{5-15}$$

  The last YC synchronization times are weighted by λ_{sum} and the new variable that is introduced for the maximum YC synchronization time now becomes:

$$t_m \geq \hat{x}_{n+1}(k+N_p) \oplus \dots \oplus \hat{x}_{n+M}(k+N_p). \tag{5-16}$$

- Instead of considering the minimization of the makespan, which is financially profitable, it is also possible to minimize the fuel consumption (financial and ecological lucrative). This objective can be formulated by the AGV travel distance:

$$t_d = \tau_v(k) \otimes ... \otimes \tau_v(k + N_p), \tag{5-17}$$

were $\tau_v(k)$ the selected travel time for $\hat{x}(k)$. New inequality constraints are introduced, since all τ_v are not directly expressed in the equations. The new inequalities become:

$$\hat{x}_d(k) \geq \tau_v(k) \otimes a(k) \otimes c(k), \tag{5-18}$$

were $\tau_v(k)$ corresponds to the time between $a(k)$ (the assigned QC) and $c(k)$ (the crane at which the selected AGV terminated its previous cycle). Only the travel distance **to** the container job is taken into account, since the travel distance **between** the cranes in a container job is not variable. The minimization of purely the travel distance then becomes:

$$\min_{z,t_d} J = \min_{z,t_d} t_d. \tag{5-19}$$

This distance objective is weighted to the maximum QC synchronization time in the ratios 1:1 and 1:10.

It should however be noted that the minimization of the objective functions cannot be implemented exactly the same as the theory prescribes. This due to the formulation of the model equations. These describe from which time on a synchronization **can** take place by means of the inequality sign, which means that it might happen later. Only due to the minimization, it is ensured that the fastest synchronization time possible is used. Therefore, all the elements of $\mathbf{x}(k)$ should at least have very little weight in the minimization, regardless if they belong to the theoretical objective function or not.

5-1-3 Search algorithms

Before the results are discussed, the used solvers are reviewed quickly. At first, five different solvers are considered:

- the standard integer linear programming (ILP) solver of MATLAB,
- the GNU linear programming kit (GLPK) solver,
- the coin-or branch and cut (CBC) solver,
- the Gurobi solver and
- the CPLEX Tomlab solver,

To solve the MILP problem, most modern solvers use the same methodology. They start by preprocessing the problem by *presolving* the problem and using *heuristics*. The MILP problem is solved by a *branch-and-bound* algorithm. When the solver applies *cutting planes* during the branch-and-bound algorithm, it solves the problem with a *branch-and-cut* algorithm [37, 38]. The interested reader is referred to Appendix F for more information on the branch-and-cut algorithm, the presolving and heuristics.

5-2 Benders' decomposition

Another approach to obtain an optimal schedule is to use Benders' decomposition [39]. This method also uses cutting planes, only now they need to be defined explicitly. They can be generated by a cut generation linear program (CGLP) which has a feasible solution that defines a family of inequality constraints [40]. Suppose in general, all continuous variables are called $\mathbf{x}$ and all integer (or binary) variables are called $\mathbf{z}$. The Benders' decomposition converts a MILP with continuous variables $\mathbf{x}$ and integer variables $\mathbf{z}$ to a problem only involving integer variable $\mathbf{x}$ and a single continuous variable η. Here, η describes the contribution of the continuous variables in the objective function, thus $\eta = c_{obj,x}^T \mathbf{x}$. This problem is called the **master problem** and solves the primal problem with the relaxation in the $(\mathbf{z}, \eta)$ space. This temporary optimal solution $(\mathbf{z}^*, \eta^*)$ is sent to the **dual slave problem**, where is $\mathbf{z}^*$ the temporary optimal AGV assignment and η^* a temporary optimal cost. The dual slave problem is linear and finds the optimum $\mathbf{x}^*$ such that $(\mathbf{x}^*, \mathbf{z}^*)$ is feasible for the original problem and such that $\eta^* = c_{obj,x}^T \mathbf{x}^*$. When the objective of the dual slave problem does not coincide with η^*, an **optimality cut** is added to the master problem. When the objective equals η^*, the optimum has been found and the algorithm terminates.

In some systems it is possible that the dual slave problem is infeasible. Instead of a Benders' optimality cut, a Benders' feasibility cut is added to the master problem. Since in the defined container terminal system feasibility of the dual slave problem will not be an issue (it is supposed that there is always enough time available), this latter cut will not be considered.

The Benders' decomposition is not often used when solving MILP problems. However, it is a commonly applied method in stochastic programming, which is to be explained in Chapter 7. The advantage is the decomposition between the integer and real variables. Since the stochastic variables will be introduced in the dual slave problem, which is an LP problem, the multiple realizations can be solved fast.

5-2-1 Benders' cuts

The standard MILP problem is given in terms of previous named matrices on the left side. At the right side, the matrices are reformulated such that the notation stays clear.

$$\min_{x,z,t} c_{obj,x}^T \mathbf{x}(k) + c_{obj,z}^T \mathbf{z}(k) \qquad \min_{x,z,t} c_{obj,x}^T \mathbf{x}(k) + c_{obj,z}^T \mathbf{z}(k)$$

subjected to

$$\begin{aligned} -\beta \mathbf{F_1 z} &\geq \mathbf{B_1 \Theta_1} + \beta \mathbf{K_1 w} & \mathbf{F_1 z} &\geq \mathbf{b} \\ -\beta \mathbf{F_2 z} + \mathbf{E_2 x} &\geq \mathbf{B_2 \Theta_2} + \beta \mathbf{K_2 w} & \mathbf{F_2 z} + \mathbf{E_2 x} &\geq \mathbf{r} \\ z &\in \{0,1\} & z &\in \{0,1\} \\ x &\geq 0 & x &\geq 0 \end{aligned} \tag{5-20}$$

and will be referred to as the **original problem**. The first set of inequalities contains all constraints on solely the binary variables z. The second set contains all other constraints

described in the previous section.

The Benders' decomposition states that the solution in the following problem will be equivalent to the original problem:

$$\min_{z,\eta} c_{obj,z}^T \mathbf{z}(k) + \eta$$

subjected to

$$\begin{aligned} \mathbf{F_1 z} &\geq \mathbf{b} \\ \eta &\geq \bar{\pi}^T \left(\mathbf{r} - \mathbf{F_2 z}\right) \\ z &\in \{0,1\} \end{aligned} \tag{5-21}$$

and is denoted as the **primal problem**. As seen before, η takes into account $c_{obj,x}^T \mathbf{z}(k)$ and set Π contains the vertices of the polyhedron D, defined by

$$\begin{aligned} \pi^T \mathbf{E_2} &\leq c_{obj,x}^T \\ \pi &\geq 0. \end{aligned} \tag{5-22}$$

An iterative solution method is applied that uses cuts to reduce the solution space. The Benders' decomposition iterates between the master and dual slave problem.

1. Solve the master problem:

   $$\min_{x,\eta} c_{obj,x}^T \mathbf{z}(k) + \eta$$

   subjected to

   $$\begin{aligned} \mathbf{F_1 z} &\geq \mathbf{b} \\ < \text{ Benders' cuts } &> \\ z &\in \{0,1\} \end{aligned} \tag{5-23}$$

   with initially no Benders' cut at all. The master problem is a relaxation of the primal problem that keeps the integrality constraints. Now let a temporarily optimal solution to the master problem be denoted by $(\mathbf{z}^*, \eta^*)$.

2. Solve the dual slave problem:

   $$\max_{\pi} \pi^T \left(\mathbf{r} - \mathbf{F_2 z}^*\right)$$

   subjected to

   $$\begin{aligned} \pi^T \mathbf{E_2} &\leq c_{obj,x}^T \\ \pi &\geq 0 \end{aligned} \tag{5-24}$$

   which is a linear problem and therefore the result can be obtained fast.

3. Let the optimal of the maximization in formulation (5-24) be denoted as ζ^* and the optimal vertex as $\bar{\pi}$. If $\zeta^* \leq \eta^*$, the algorithm terminates, since the solution is feasible and optimal for the primal problem. Otherwise, the Benders' optimality cut is added to the current master problem and another iteration is initiated. The Benders' optimality cut:

$$\eta \geq \bar{\pi}^T (\mathbf{r} - \mathbf{F_2 z}) . \tag{5-25}$$

If the maximization of the dual slave problem and the Benders' cut are examined more closely, it becomes clear that only the constraints that are crucial and determinative for the synchronization times are penalized in the optimality vector $\bar{\pi}$ per subproblem. It then becomes clear that the master problem chooses different schedules such that η will remain zero. Therefore, most (as not all) feasible schedules are searched without the lower bound η increasing. Once it cannot find a costless schedule anymore due to the Benders' cuts, the algorithm finds the optimal solution fast, since it already gathered all necessary constraints. A typical evolution of the lower bound η and optimal solution (new try upper bound) ζ^* can be seen in Figure 5-2.

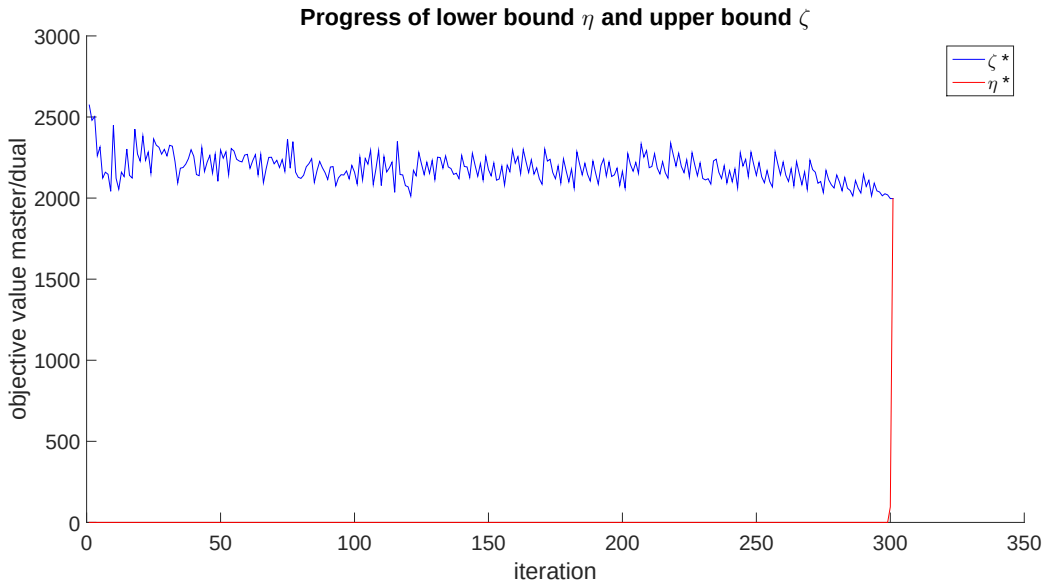

Figure 5-2: Typical progress of lower bound and temporarily solution of the classical Benders' decomposition.

Castillo et al. (2006) [41] proposed a different Benders' cut. They suppose that not only the continuous variables are variable in the subproblem, but also the integers. However, by fixing the integers by the constraints $\mathbf{z} = \mathbf{z}^*$, they can be treated as continuous. The dual problem and subproblem are defined slightly different [42]. They propose the Benders' cut:

$$\eta \geq c_{obj,x}^T \mathbf{x} + \lambda_{max}(\mathbf{z} - \mathbf{z}^*). \tag{5-26}$$

This Benders' cut "remembers" the cost $c_{obj,x}^T \mathbf{x}$ for the schedule $\mathbf{z}^*$. In the system of the container terminal the cut does not solve the earlier denoted problem. However, keeping these cuts in mind, a new proposal for an algorithm can be done. The algorithm considers

the jobs one by one, adding a cost to the calculated schedules. Since these will be included in the optimization for the next job, the search space will be reduced considerably. The master problem now becomes:

$$\min c_{obj,z}^{1\ T}\mathbf{z}^1 + \eta^1$$

subjected to

$$\mathbf{F_1}^1\mathbf{z}^1 \geq \mathbf{b}^1$$
$$\eta^1 \geq \pi^{1\ T}(\mathbf{r}^1 - \mathbf{F_2}^1\mathbf{z}^1).$$

Here only the first job is calculated and the cuts are retrieved by the classical dual slave problem. Additionally, the cost is calculated by a linear optimization, deriving $c_{obj,x}^1\mathbf{x}^1$ and $\iota_i = 1$ when $\mathbf{z}_i^* = 1$ and $\iota_i = -1$ when $\mathbf{z}_i^* = 0$, obtaining:

$$\nu^1 \geq c_{obj,z}^{1\ T}\mathbf{z}^{*\ 1} - \iota^{1\ T}(\mathbf{z}^1 - \mathbf{z}^{*\ 1}).$$

The second job:

$$\min c_{obj,z}^{2\ T}\mathbf{z}^2 + \eta^2 + \nu^1$$

subjected to

$$\mathbf{F_1}^2\mathbf{z}^2 \geq \mathbf{b}^2$$
$$\eta^2 \geq \pi^{2\ T}(\mathbf{r}^2 - \mathbf{F_2}^2\mathbf{z}^2)$$
$$\nu^1 \geq c_{obj,z}^{1\ T}\mathbf{z}^{*\ 1} - \iota^{1\ T}(\mathbf{z}^1 - \mathbf{z}^{*\ 1}).$$

Where the problem is exactly the same, but formulated for two jobs. The additional variable ν introduces a cost for the first job assignment. Again, a cost is calculated for the first two jobs $c_{obj,x}^2\mathbf{x}^2$ and corresponding ι^2. This creates ν^2, which will be used instead of ν^1 in the optimization for the next job.

5-3 Results of the SMPL model - MILP approach

The MILP problem was implemented in MATLAB 2014b (64-bit version) and the computations are done on a computer using the Windows 7 Professional 64-bit operating system. It has 4 Intel® Core™ i5-3570 processors with a clock speed of 3.4 GHz. The random access memory (RAM) of the computer is 8192 MB, while MATLAB has a memory usage of at most 8735 MB.

The solution to the MILP problem returns the cost of the optimization, highly depending on the handling times that are used in the optimization. The AGVs can be tracked and the QC waiting times calculated, but can differ considerably when the handling times and AGV travel times are adjusted. The handling times are shown in Table 5-3.

The settings that are used in computing the results are given in Table 5-4, unless otherwise indicated.

$$\begin{aligned} \tau_2(k) = \tau_2 &= 80, & \tau_5 &= 15, \\ \tau_3 &= 10, & \tau_6 &= 160. \end{aligned}$$

Table 5-3: Handling times of the QCs and YCs, with τ as defined in Section 4-1-2.

tc	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
N	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10
M	3	4	5	5	6	7	8	8	9	10	11	11	12	13	14	14	15	20
V	3	6	9	12	15	15	18	21	24	27	30	30	33	36	39	42	45	50
P	5	10	15	20	25	25	30	35	40	45	50	50	55	60	65	70	75	85
N_p	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

Table 5-4: The specifications of the test cases used in the experimental runs, where tc the test case, N the number of QCs, M the number of YCs, V the number of AGVs, P the previous considered cycles and N_p the number of container jobs on the horizon.

The AGV travel times depend on the layout of the terminal, which is chosen parallel and with the distances as in Figure 5-3 and an average speed of 2.5m/s, taken into account accelerations and delays. The work queue is presented in Table 5-5 (unless otherwise indicated).

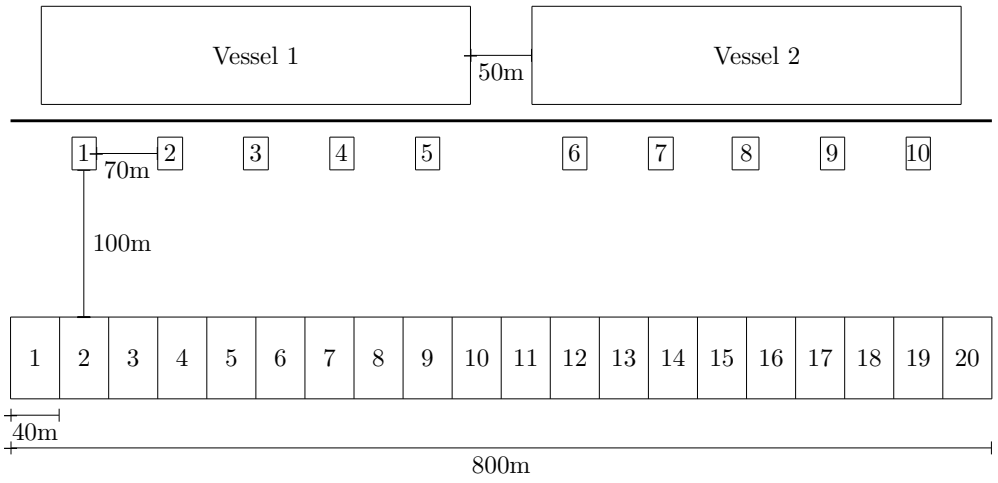

Figure 5-3: Container terminal layout for all the test cases, unless otherwise indicated.

container jobs:	1 to c_j	e.g.	$\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix}^T$
QC work queue:	repeat 1 to N	e.g.	$\begin{bmatrix}1 & 2 & 1 & 2 & 1\end{bmatrix}^T$
YC work queue:	repeat 1 to M	e.g.	$\begin{bmatrix}1 & 2 & 3 & 1 & 2\end{bmatrix}^T$
cycle type	discharge jobs d_1	e.g.	$\begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}^T$

Table 5-5: Work queue for all the test cases unless otherwise indicated.

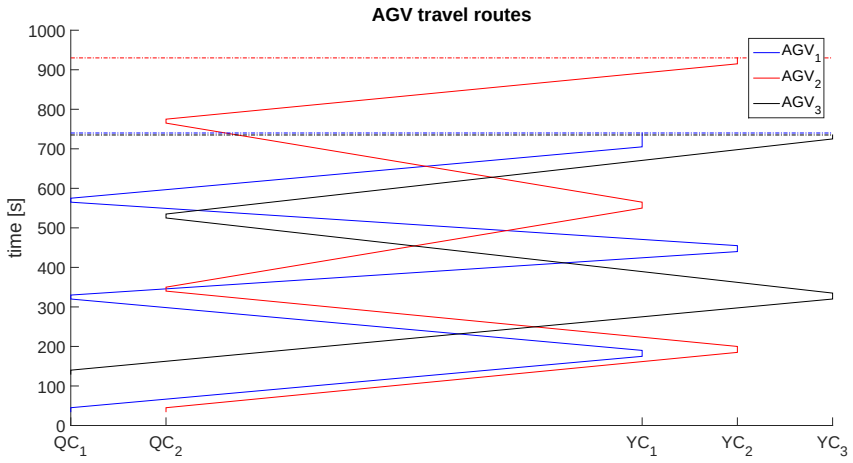

Figure 5-4: The AGV routes resulting from solving the system described in Appendix G.

First, some simple schedules are produced and their correctness verified by hand calculations, of which an example is provided in Appendix G. In Figure 5-4 it is shown how the AGVs move through the system in time. Next, the MILP formulation is considered ("the MILP approach"). With this algorithm, an optimal schedule for all the test cases is derived with the parameters given above. The following can be established:

- how the number of constraints increases with the complexity of the problem,
- how the number of variables increases with the complexity of the problem,
- which solver provides the fastest solution time,
- how the solution time is affected by the complexity of the problem,
- whether different objective functions return different objective values.

When these aspects have been considered, the model and the optimal schedules are evaluated by varying and analyzing the effect of the parameters

- N_p: the number of container jobs,
- V: the number of AGVs,
- P: the number of previous cycles that are taken into account,
- work queue: mixing discharge and loading jobs.

5-3-1 Analysis MILP approach - terminal size, solvers and objective functions

One of the goals in this thesis is to design a scheduling algorithm that can be used on-line. Therefore it needs to be fast. The speed of the optimization depends mainly on the size of the problem and the applied solvers.

Terminal size

Since the max-plus model describes every path in the system, the complexity grows rapidly as the problem size increases. Two meaningfull benchmarks are the number of constraints and the variables that need to be optimized. When the test cases defined earlier are used, it becomes clear that the number of constraints grows quadratic in the number of QCs N and YCs M. Even though an increasing number of AGVs does not directly influence the number of constraints (as it does not appear in Eq. (5-4)), it determines how many previous cycles should be taken into account. It therefore determines the magnitude of P, which causes the number of constraints to increase rapidly. The number of variables grows linear with respect to the number of QCs, YCs and P, as becomes clear with the definition of vector $\mathbf{z}$ in Section 5-1. This is illustrated in Figure 5-5.

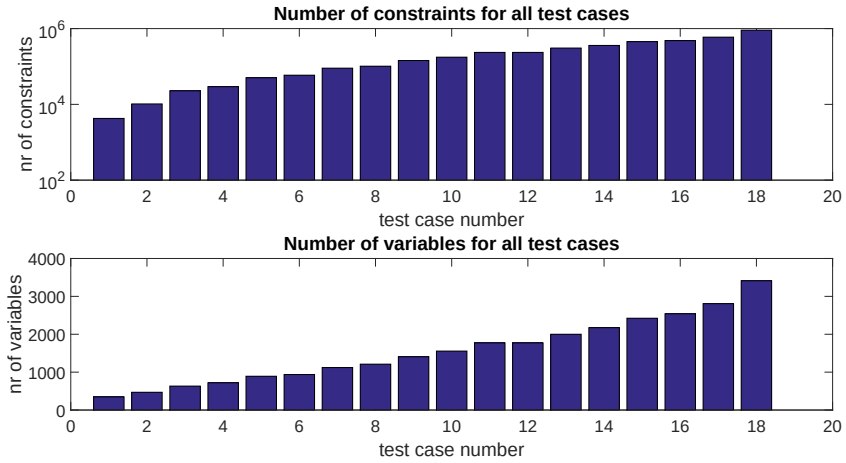

Figure 5-5: The number of constraints and variables of the eighteen test cases with increasing terminal complexity.

Solvers

Many different solvers can be used to obtain the optimal schedule. Five different solvers are tested for their computation speed. The GLPK solver already takes over five minutes to solve the second test case. The standard MATLAB MILP solver and CBC solvers also perform not nearly as good as the Gurobi and CPLEX (Tomlab) solvers, as can be seen in Figure 5-6. Whenever there is no solution time drawn for a solver, it means that the solution time was over five minutes.

The only two solvers that are able to obtain an optimal schedule for the largest test case within the deadline of five minutes are the Gurobi and CPLEX solvers. Both the solvers return the same objective values for all test cases, implying that they found the true optimum of the problem.

From now on, all optimizations will be done using the Gurobi solver, since it performs best. The CPLEX solver performs approximately just as good, but requires a license, while the

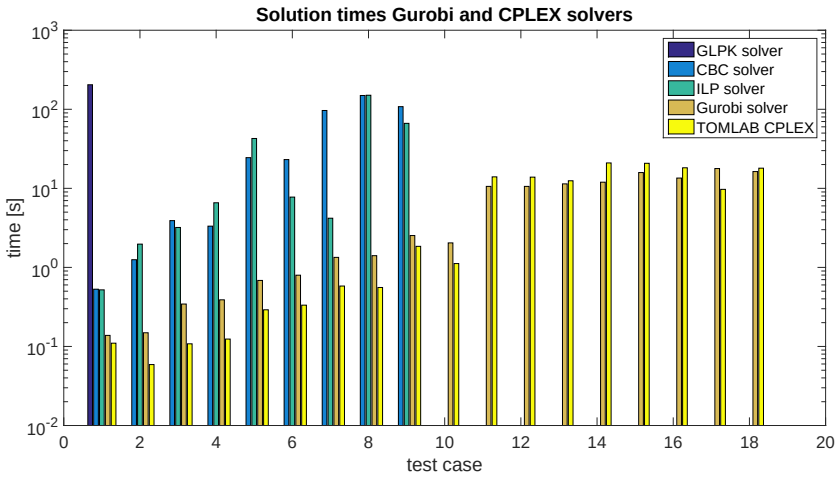

Figure 5-6: The performances of the different solvers. The upper plot shows the performance of the slower GLPK, CBC and standard MATLAB solvers. The lower plot shows the results of the faster CPLEX and Gurobi solvers. Solution time increases when the terminal size increases.

Gurobi solver provides a free academic license. It should be noted that the best performing open-source solver is the CBC solver.

Objective function

Interesting to see is if and how the performance index influences the results. In Section 5-1-2 the different objective functions are explained. First, let us look at minimizing the last performing QC ("max minimization") versus the sum of all the last synchronizations of the QCs ("sum minimization") (Eq. (5-13), $J_{max} : \lambda_{max} = 1,\ \lambda_{sum} = 0$ and $J_{sum} : \lambda_{max} = 0,\ \lambda_{sum} = 1$).

The results for the test cases appear to be exactly the same, excluding test cases 13 and 16. Figure 5-7 shows that QC_5 in test case 13 and QC_6 in test case 16 have different synchronization times when the performance index is adjusted from max minimization to the sum minimization. As expected, the summation has the best overall performance. In this objective, all last QC synchronizations have the same weight in the optimization, in contrast to minimizing the maximum, where only one QC has priority (Eq. (5-14)).

However, when more container jobs are scheduled in one single simulation, the differences between QC synchronization times become larger. The priorities of the objective functions distinguish themselves as more choices become available in a simulation, of which an example is given for test case 4 in Table 5-6. The ratio in the last column is that of the average synchronization time to the maximum synchronization time. The highest ratio is that of the performance index minimizing the maximum QC time. This objective suppresses the maximum, but compensates by increasing other QC synchronization times. Another important aspect is that of the computation time. In Figure 5-8 it can be seen for several test cases that there is a considerable difference in computation time.

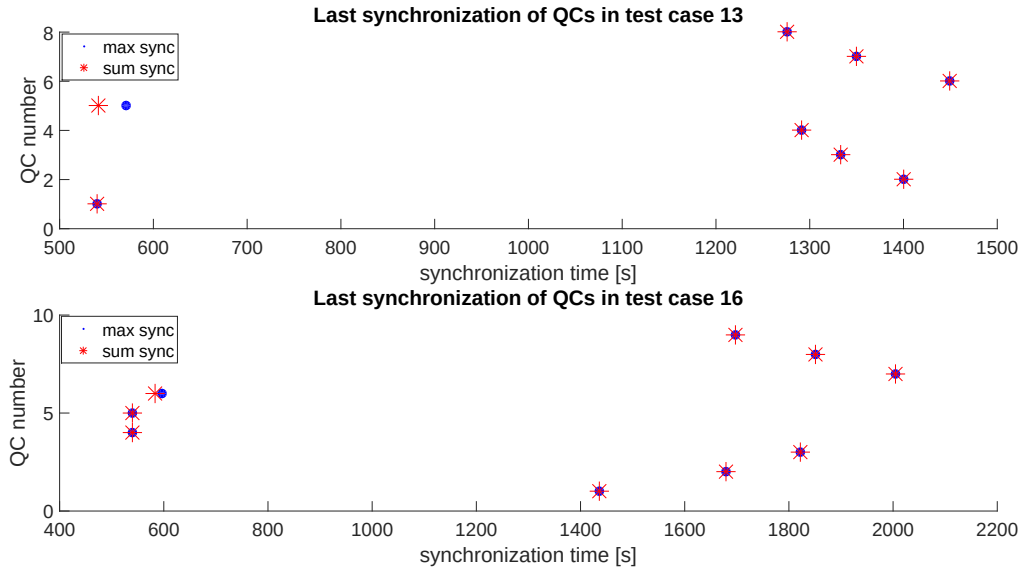

Figure 5-7: The last synchronization times of all the QCs for test cases 13 and 16 when the max minimization and sum minimization are implemented as performance index.

test case 4 ($N_p = 15$)

	QC_1	QC_2	QC_3	ratio
sum	810	810	828	0.986
max	810	810	828	0.986
trade-off	810	810	828	0.986

test case 4 ($N_p = 30$)

	QC_1	QC_2	QC_3	ratio
sum	1260	1318	1360	0.965
max	1290	1332	1318	0.979
trade-off	1260	1318	1360	0.965

Table 5-6: The last synchronizations of the QCs in test case 4 using three performance indices (sum, max and $\lambda_{max} = 0.5$). When the prediction horizon increases, the difference becomes visible.

The different tested objective functions that minimize the YC synchronization times also show almost exactly the same results in the test cases. None of the differences effect the makespan. More jobs will most probably result in larger differences, as is seen for the QC objectives. When the QC objective is compared to the YC objective, the QC objective performs better. In eleven of the 90 cases (18 test cases $\times$ 5 trade-offs) it obtained faster synchronizations up to 155 seconds, but none of them influencing the makespan.

A last objective function describes the minimization of the travel distance of AGVs, deriving very different results. The computation time increases drastically, far exceeding 300 seconds for the largest case, while also the makespan is often much larger when only the travel distance is considered, which is shown in Table 5-7. The performed simulations use $\lambda_{max} = 1$, $\lambda_{sum} = 1$ and the weight on minimizing distance t_d: $\lambda_d = \{1, 10\}$. All results can be seen in Appendix **??**. It is concluded that this objective function is not preferable to that of minimizing the QC or YC synchronization times.

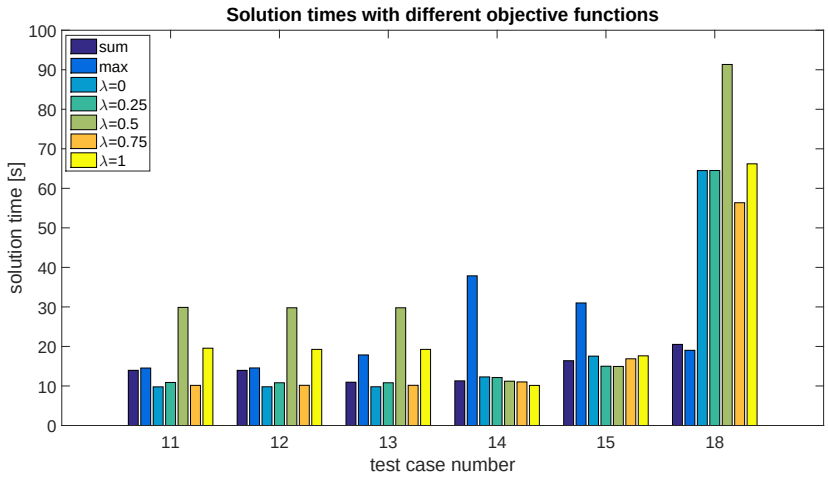

Figure 5-8: The solution computation times for test cases 11 to 15 and 18.

t.c	$\omega_d = 1$	$\omega_d = 10$
1	1%	6%
4	0%	37%
8	6%	21%

Table 5-7: The percentage loss when the travel distance objective is compared to the QC objective function.

5-3-2 Analysis MILP approach - varying parameters

The effect of varying the number of jobs (horizon), AGVs and P is studied. Lastly, also the difference in cycle types will be considered.

Scheduling horizon

To determine how many jobs should be scheduled, a trade-off should be made between the solution computation time and how far ahead in time one wants to schedule. Due to the perturbation in the container terminal system, it will not make sense to schedule 2 hours ahead. However, to obtain some overall optimum, it will be necessary to extend the number of jobs to a certain horizon. Table 5-8 shows the number of jobs that are tested N_p and the scheduled horizon in seconds, the solution computation time t_{sol}, how long it takes on average for the next job is ready at the quay side j^t (based on 90s cycle time of QC) and the fraction of jobs per QC j^f.

There are different approaches to choose the horizon. It is possible to compute as much as possible jobs within a certain time, but a better approach might be to schedule a number of jobs for every QC. A minimum amount would be two jobs per QC, preferably more. However, there is a maximum amount that should be scheduled, which depends on:

1. whether the computation time exceeds the scheduled time,

N	j^t [s]	N_p	j^f	$t_{sol}[s]$	N_p	j^f	$t_{sol}[s]$	N_p	j^f	$t_{sol}[s]$
2	45	5 = 225s	2.5	0.02	10 = 450s	5	0.23	15 = 675s	7.5	0.19
3	30	5 = 150s	1.7	0.08	10 = 300s	3.33	0.41	15 = 450s	5	0.34
4	22.5	5 = 113s	1.25	0.08	10 = 225s	2.5	0.28	15 = 338s	3.75	0.68
5	18	5 = 90s	1	0.22	10 = 180s	2	0.51	15 = 270s	3	1.39
6	15	5 = 75s	0.83	0.14	10 = 150s	1.67	2.73	15 = 225s	2.5	16.4
7	12.9	5 = 64.3s	0.71	0.90	10 = 129s	1.43	4.00	15 = 193s	2.14	20.3
8	11.25	5 = 56.3s	0.63	0.53	10 = 113s	1.25	7.54	15 = 169s	1.88	20.9
9	10	5 = 50s	0.56	1.99	10 = 100s	1.11	15.25	15 = 150s	1.67	33.4
10	9	5 = 45s	0.5	2.91	10 = 90s	1	16.88	15 = 135s	1.5	69.6

N	j^t [s]	N_p	j^f	$t_{sol}[s]$	N_p	j^f	$t_{sol}[s]$
2	45	20 = 900s	10	0.49	25 = 1125s	12.5	0.66
3	30	20 = 600s	6.67	0.61	25 = 750s	8.33	1.00
4	22.5	20 = 450s	5	1.35	25 = 563s	6.25	1.91
5	18	20 = 360s	4	2.62	25 = 450s	5	3.72
6	15	20 = 300s	3.33	26.9	25 = 375s	4.17	28.3
7	12.9	20 = 257s	2.86	36.0	25 = 322s	3.57	54.1
8	11.25	20 = 225s	2.5	45.7	25 = 281s	3.13	90.9
9	10	20 = 200s	2.22	66.36	25 = 250s	2.78	108
10	9	20 = 180s	2	87.19	25 = 225s	2.5	136

Table 5-8: Table that describes the horizon per number of QCs. The red number indicate that an AGV is scheduled for approximately every next 2 to 3 jobs per QC. Test cases were 1-3-5-7-9-11-13-15 and 17.

2. whether the scheduled time exceeds a certain point in time that is crucial for the reliability of the schedule due to uncertainties in the system (0.5 hour) or

3. whether the hardware of the system that runs these simulations or the MATLAB memory capacity are unable to handle the complexity.

tc	N_p	limit	tc	N_h	limit	tc	N_h	limit
1	40	2	7	100	2	13	40	1 & 3
2	40	2	8	100	2	14	30	1 & 3
3	60	2	9	90	1 & 3	15	20	1 & 3
4	60	2	10	70	1 & 3	16	20	1 & 3
5	80	2	11	50	1 & 3	17	20	1 & 3
6	80	2	12	50	1 & 3	18	20	1 & 3

Table 5-9: The maximum horizon per test case and the nature of its limitation.

The minimum horizons are marked red in Table 5-8. The maximum horizon for test cases 1 to 8 is scheduling half an hour (limitation 2). For test cases 9 to 18 this is not possible due to memory issues and too long computation times. The maximum number of scheduled jobs can be seen in Table 5-9, with the limitations from the above summation. It should be pointed out that in all test cases initially half an hour was scheduled, causing memory issues in test

cases 9 to 18 (limitation 3). A horizon just within the capacity of MATLAB resulted in too long computation times (limitation 1).

Number of AGVs

It is expected that the number of AGVs strongly influences the QC waiting times, as seen in Section 4-1-1. Therefore, one AGV per QC will cause enormous QC waiting times, while four AGVs are more than enough to serve the QCs on time (optimal performance assumed). This is exactly what results from simulations with six new test cases, where $N = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$ and where the number of AGVs equals: $\begin{bmatrix} N & 2N & 3N & 4N \end{bmatrix}$ and $M = 3N$, $P = 2V$ and $N_p = 4N$.

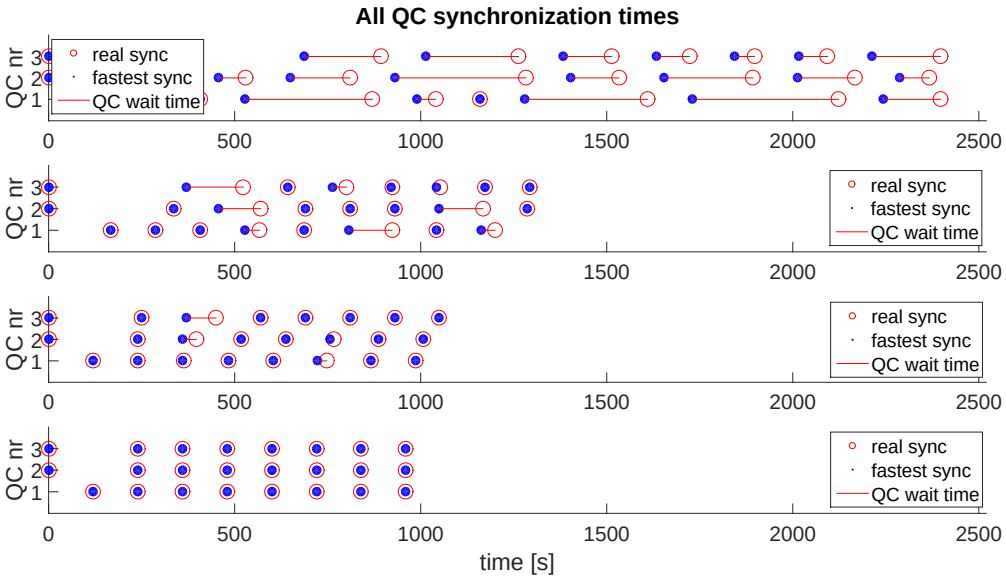

Figure 5-9: A typical representation of all the last QC synchronizations for an increasing number of AGVs (from top to bottem: 3-6-9-12 AGVs).

To exclude the possibility that the horizon is too short to notice any delays, the simulations were repeated with $N_p = 8N$. Lastly, to also exclude any dependence on the repetitive sequence of the work queue, the YC sequence is randomized. A typical result of increasing the number of AGVs can be seen in Figure 5-9, were the QC finished its last job in the blue dots and synchronizes at the red circles. The red lines are the QC wait times, which reduce when the number of AGVs increases.

Number of previous cycles

To study the number of previous cycles, the same test cases are used as in the evaluation of the number of AGVs (with the number of AGVs equal to $2N$). Analyzing the results leads to the conclusion that considering more than $1.5 \times V$ cycles back does not enhance the performance any more. It would however increase the model complexity and hence the

computation time. An example of a result can be seen in Figure 5-10, where there are four AGVs and the synchronization times of the two QCs are shown. When $P < 1.6$ is chosen, the optimization loses freedom in the scheduling of the AGVs and the performance becomes worse. As an explanation, suppose $P = V$ and the initial scheduling sequence for three container jobs is: AGV_1 - AGV_2 - AGV_3. The constraint Eq. (5-10) combined with Eq. (5-6) ensures that every AGV, every path through the system continues, except when it terminates at a sink node. Assuming that none of the AGVs terminate at a sink node, the next jobs have to be performed in the repetitive sequence of AGV_1 - AGV_2 - AGV_3. This will result in an unfavorable schedule.

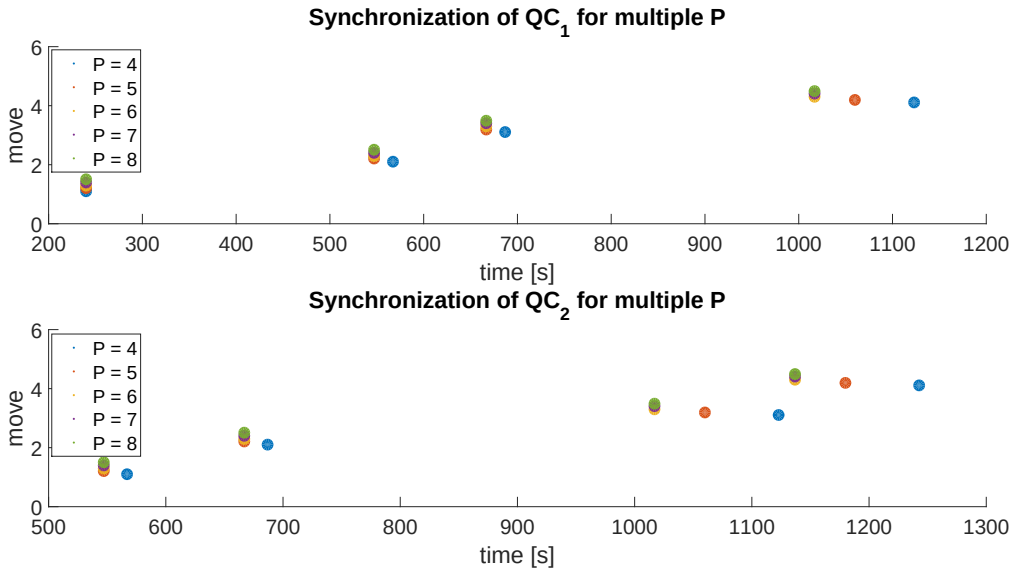

Figure 5-10: The synchronization times when P is varied. When $P = 4 = V$ and when $P = 5 = 1.25V$ there are many delays, which are completely resolved when $P \geq 6 = 1.5V$.

Cycle types

When comparing the parameters, only discharge jobs are considered. It is however also possible to mix the discharge jobs with loading jobs. As can be seen in Figure 5-11, this causes delay in the QC synchronizations. In this particular test case, which is representative for all the simulations, the following parameters are used:

$$N = 3,\ M = 9,\ V = 9 \text{ and } P = 18.$$

Even if the number of AGVs is increased, the delay still remains. This can be explained by considering the definition of the model, which does not allow any job order switching. The importance and impact of this limitation will be explained in the next section.

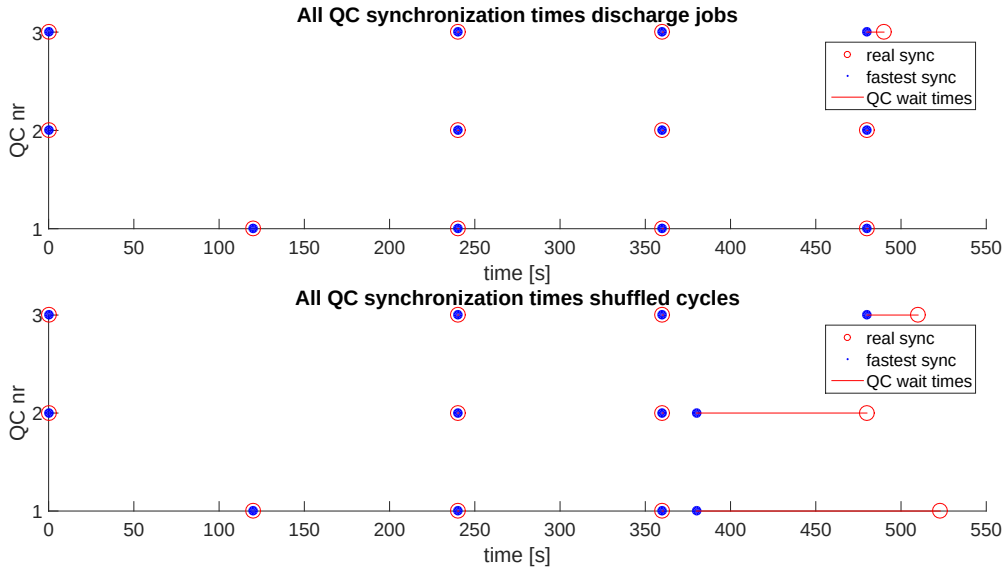

Figure 5-11: The QC synchronization times for only discharge jobs and for shuffled cycle types.

5-3-3 Switching job orders

In the definition of the model, it is not possible to switch the job order, since Eq. (4-18) shows:

$$x(k) \geq \mathbf{A}_0 \otimes x(k) \oplus \mathbf{A}_1 \otimes x(k-1) \oplus \mathbf{A}_2 \otimes x(k-2) \oplus ... \oplus \mathbf{A}_P \otimes x(k-P)$$

hence it always holds that $x(k) \geq \{x(k-1),\ x(k-2),\ ...,\ x(k-P)\}$.

Delay type I − YC
Now suppose there is a discharge job $(k+7)$ which has to be stacked in the yard by YC_2. The container is expected to arrive at time $t_7 = 1500$s, while the crane itself is ready at $t_0 = 900$s. The next job assigned to YC_2 is the discharge job $(k+11)$, arriving at the yard at $t_{11} = 1000\text{s} < t_7$. Now the YC will wait with performing $(k+11)$ until it finishes $(k+7)$ (as Eq. (4-18) prescribes), even though it easily can perform $(k+11)$ without losing time in job $(k+7)$. These situations cause delay at the yard side, occupying an AGV for an unnecessary long time. Important is to realize that this flaw is in the model, whereby it loses accuracy in describing the dynamics of the system, since in a container terminal, the YC probably decides to perform job $(k+11)$ first.

Delay type II − QC
A second delay that can occur is at the quay side. From the simulations (Figure 5-11) it is clear that when a QC performs a loading job preceded by a discharge job (thus cycle type 2), the discharge job has a slow synchronization. Apparently, the AGV lags the fastest QC synchronization time, concluding that the AGV starts traveling too late. To solve this issue, the YC-AGV synchronization should happen earlier, implying that the delayed job should move up in the work queue of the YC. This might result in extra waiting time for an AGV

performing another job, but it should be kept in mind that delay for a QC is the most unfavorable situation in the container terminal.

There are two courses of action that can be taken. A full switching scheme could be implemented, where new constraints are defined and new binary variables introduced for all job switches that are allowed. These are then directly introduced into the MILP problem, increasing the number of constraints and variables tremendously. Another method would be to only allow switching when the predicted schedule shows a long waiting time for a QC or YC. In this threshold switching, recalculation is needed to determine if a job switch improves the schedule. The switching of jobs is highly recommended, but not further explored in this work. More information on the switching schemes is provided in Appendix H.

5-4 Results of the SMPL model - Benders' decomposition approach

Figure 5-2 (Section 5-2) already explained the problem when the classical Benders' decomposition is implemented. Due to the relaxation of the constraints, the Master problem is able to find an unconstrained schedule most (or all) of the time. Hereby, there are too many iterations that need to be computed by the algorithm to obtain an optimum in as little time as the MILP approach. Alternative Benders' cuts are proposed.

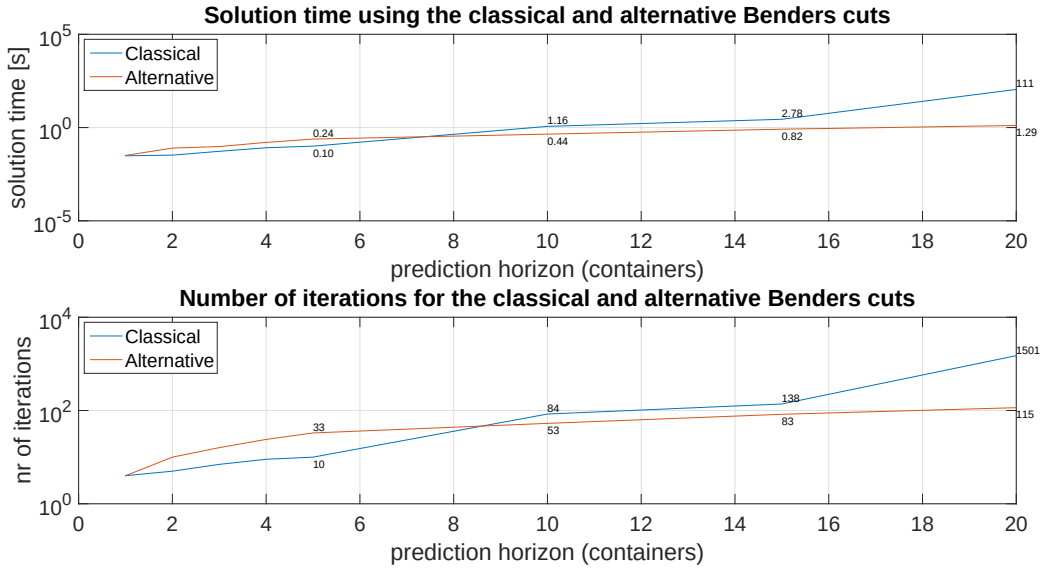

Figure 5-12: The difference regarding the solution times and number of iterations between the classical and alternative Benders' cuts for a simple case of one QC, two YCs and two AGVs.

Figure 5-12 shows the results when there are only one QC, two YCs, two AGVs and $P = 3$. For very little jobs, approximately 7, the classical Benders' cuts perform better. However, when the number of container jobs increases, the alternative cuts perform much better. For

example. for 20 jobs it takes 1.29s, while the classical cuts result in 111s. The number of iterations progresses in the same manner as the solution time, since they are related. Again, the alternative cuts perform worse when using a very small prediction horizon, but better than the classical cuts when the horizon increases.

From these results it is concluded that the alternative cuts might reduce the problem, but they do not solve it. Trying larger cases, such as test case 1 (defined in Page 43) with $N_p = 10$ container jobs results in a computation time of more than 1.5 hours. The method performs thus not nearly as good as the MILP approach, which solved the largest test cases within 20s. The Benders' decomposition will not be considered any further as a solution algorithm.

Lastly, it should be noted that the correctness of the implementation of the Benders' decomposition is checked by comparing the objective and resulting schedules to those of the MILP approach. Since they return the same schedules, the correctness of the implementation of the Benders' decomposition approach is verified.

Chapter 6

Model Predictive Scheduling

In the previous chapter, it is explained that the schedule is not reliable when a large horizon is chosen. Since an on-line scheduling method is desired, rescheduling is necessary. To obtain this, model predictive scheduling (MPS) is used, which in the container terminal system has several important advantages [21, 43].

- The computation time might increase such that it cancels out the time gained by the optimization or even deteriorate it. MPS uses a receding horizon principle to avoid this negative effect on the computation time.
- Moreover, it can use a part of the previous solution as an initial guess to the new optimization, deriving the optimal schedule faster in the new optimization than in the initial problem.
- It handles constraints in a systematic way during the design and implementation of the controller.
- It enables reactive operational scheduling. Based on a model, it is possible to optimize the performance by rescheduling using **new state information**, which makes the schedule more accurate.

In the MPS method there are five important parts recognizable that coincide with those in MPC, namely a process (and potentially a disturbance) model, the performance index, the constraints, the optimization and the receding horizon principle. Only the latter needs to be added to the existing SMPL model.

6-1 MPS for the SMPL model

When MPS is applied, a receding horizon is added to the problem. This means that the optimal schedule for a prediction horizon of N_p jobs is calculated, where after the horizon

is shifted N_r event counters k and a new schedule is retrieved. When the schedules are obtained off-line, the shifting horizon eases the computational burden, since smaller horizons can be used to schedule the same amount of jobs. When the SMPL-MPS model runs on-line, information on the state (QC and YC synchronization times) can be obtained and used to predict a more accurate schedule. In this case, only the first N_r AGV job assignments are implemented (Figure 6-1). Subsequently, the horizon is shifted, new state information is obtained and the schedule is recalculated. Due to the shift in horizon, the same number of jobs are scheduled for every recalculation (excluding the recalculations restricted by the finite amount of containers at the end of the overall schedule).

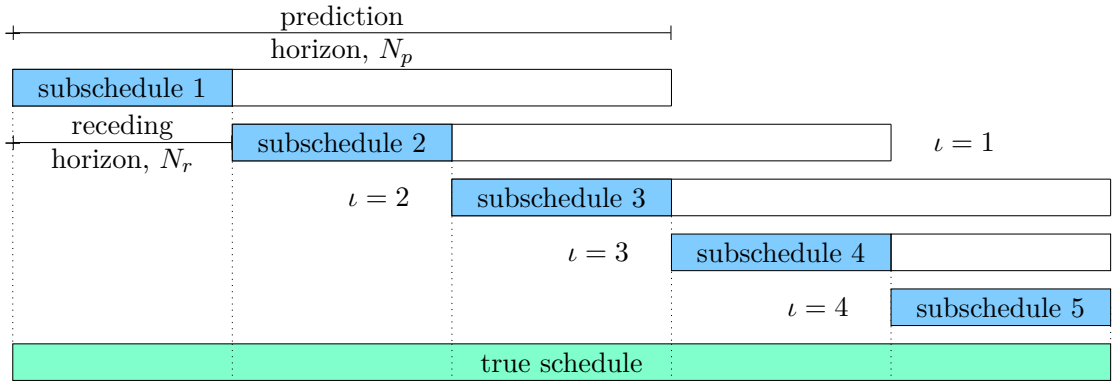

Figure 6-1: Receding horizon principle. The implemented schedule is represented in green, merged from the blue calculated schedules.

The SMPL-MPS system is described as:

$$\mathbf{x}(\boldsymbol{\kappa}) = \mathbf{A} \otimes \mathbf{x}(\boldsymbol{\kappa} - 1). \tag{6-1}$$

This model introduces the new event counter $\boldsymbol{\kappa}$, with which the state evolves as:

$$\mathbf{x}(\boldsymbol{\kappa} - 1) = \begin{bmatrix} \mathbf{x}(k + N_p) \\ \vdots \\ \mathbf{x}(k) \\ \vdots \\ \mathbf{x}(k - P) \end{bmatrix} \qquad \mathbf{x}(\boldsymbol{\kappa} + \iota) = \begin{bmatrix} \mathbf{x}(k + N_p + \iota N_r) \\ \vdots \\ \mathbf{x}(k + \iota N_r) \\ \vdots \\ \mathbf{x}(k - P + \iota N_r) \end{bmatrix} \tag{6-2}$$

for $\iota = 1, 2, 3, ...$ and for N_p the prediction and N_r the receding horizon. The tuning parameters of the SMPL-MPS approach are the rprediction horizon N_p, the receding or rescheduling horizon N_r and the trade-off variables λ_{sum} and λ_{max}.

6-2 Results of the SMPL-MPS approach

The following was analyzed for the off-line SMPL-MPS model:

- effect of λ on the makespan,

- the computation speed using initial guess,
- the trade-offs regarding receding and prediction horizon,
- the difference in makespan when MPS is applied and when not.

The last analysis will consider the value of the SMPL-MPS model on-line, when state information becomes available. The MILP approach is used to do the numerous simulations necessary to test the SMPL-MPS approach.

Effect λ on makespan

The same performance index is used as in Eq. (5-13):

$$\min_{x,z,t_n} J = \min_{x,z,t_n} \left(\lambda_{sum} \sum_{n=1}^{N} \hat{x}_n(k+N_p) + \lambda_{max} \max_{n\in(1,\ldots,N)} \hat{x}_n(k+N_p) \right). \tag{5-13}$$

Earlier it is seen that taking the performance index equal to the summation or to the maximum of the last QC synchronizations times makes no difference in the makespan. It is verified if the same applies to the receding horizon approach. The considered objectives are again:

1. summation of the last QC synchronization times ($\lambda_{sum} = 1,\ \lambda_{max} = 0$),
2. maximization of the last QC synchronization times ($\lambda_{sum} = 0,\ \lambda_{max} = 1$) and
3. a trade-off where the weight of the maximum time is $\lambda_{max} = 0.2 \times N$ and $\lambda_{sum} = 1$.

This time, there is a clear difference between the objective values, due to the larger horizons. The max minimization gives the fastest makespan, since it minimizes the last synchronizing QC. The trade-off performs slightly better on average and performs still very well in minimizing the makespan. This due to the implementation of minimizing the last synchronizing QC, but also suppressing all the last QC synchronization times. The summation performs best on average QC synchronization times, but much worse on the makespan, since it contains no weight to minimize the last synchronizing QC ($\lambda_{max} = 0$ for Eq. (5-13)). A trade-off between the average and maximum QC synchronization times appears to be the best objective value. A good overall performance is desired, but the maximum synchronization time should be bounded.

	QC_1	QC_2	QC_3	QC_4	QC_5	QC_6	QC_7	QC_8	QC_9	QC_{10}	average	max	ratio
sum	1463	1529	1598	1344	1410	1655	1680	1643	1669	1592	1558	1680	0.928
trade-off	1432	1520	1564	1442	1621	1529	1634	1643	1627	1633	1565	1643	0.952
max	1548	1614	1598	1617	1585	1529	1588	1599	1584	1596	1586	1617	0.981

Table 6-1: The max, sum and trade-off objective functions return different predictions for the last QC synchronization times of test case 17.

In Table 6-1 the last synchronizations of the QCs in test case 17 are presented. The last column of the table shows the fraction "average sync/maximum sync". This clarifies the explanation regarding the effect of the objective function on the system. The first objective

(summation) has the largest deviation and lowest ratio. Only the summation is minimized, while the maximum is totally disregarded ($\lambda_{sum} = 1,\ \lambda_{max} = 0$ in Eq. (5-13)). The second objective (maximization) suppresses the maximum synchronization time, but compensates on others. It therefore has the largest ratio and the smallest differences in synchronization times. Now $\lambda_{sum} << 1$ and $\lambda_{max} = 1$ in Eq. (5-13), minimizing the maximum synchronization time. The trade-off is in the middle of these two discussed objectives, as is expected. In the equation, it shows that both the average performance as well as the maximum are minimized, since $\lambda_{sum} = 1$ and $\lambda_{max} = 0.2 \times N$.

In Appendix J the results of the different objective functions for test cases 1, 2, 8, 9 and 17 are shown in a figure (Figure J-1) and in Table J-1.

Computation speed

The first optimization is exactly equal to the one performed in the previous chapter. However, in all the optimizations that follow due to the receding horizon, an initial guess can be provided to the optimization. This initial guess is the optimum of the previous optimization. Due to this, the computation speed increases about five to six times, disregarding some exceptions in which the optimization has difficulty finding an optimum (Figure 6-2).

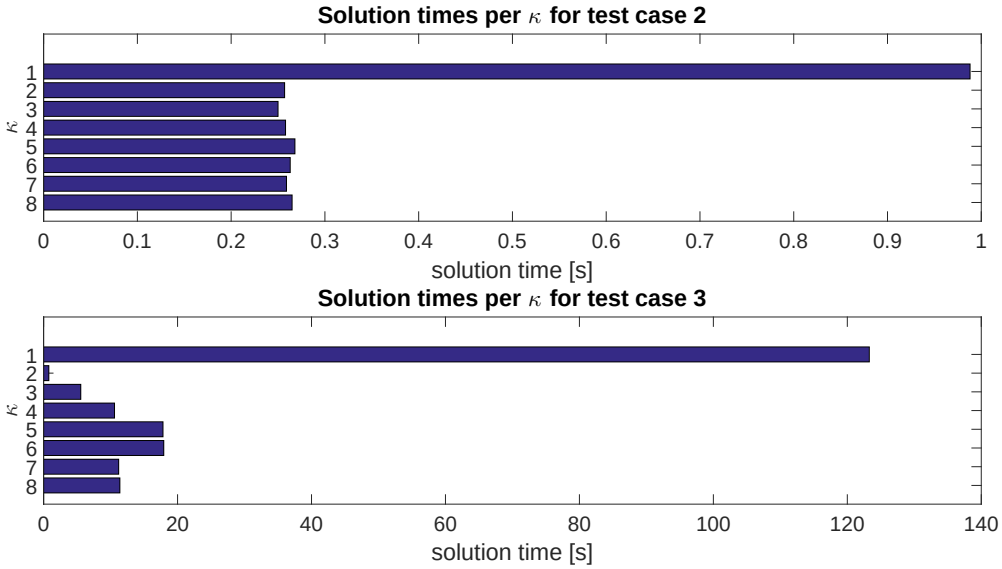

Figure 6-2: The decreasing computation time in the receding horizon approach for the test cases 2 and 3.

Trade-offs N_p and N_r

To decide which prediction horizon should be used, the same three reasons apply as did in Section 5-3-2. The prediction horizon for test cases 1 to 8 stays restricted to the maximum planning horizon of 30 minutes that is assumed. The other test cases can be calculated

faster, but they were already using almost all of MATLAB's assigned memory. Therefore, Table 5-8 still applies. It is no coincidence that the horizon does not increase. It is already stated that test cases 9 to 18 are limited by computation time, which are long since the cases are large and the computer and MATLAB run out of (RAM) memory. It is no surprise that the maximum memory capacity is almost immediately reached when increasing N_p.

Regarding N_r, if it is chosen too large, it might happen that unfavorable choices are made at the end of the schedule, which cannot be corrected any more. This is verified using different N_r in the test cases, for example test case 1, as can be seen in Figure 6-3. In this simulation 100 container jobs are scheduled and the first 40 are shown, where $N_p = 25$ and $N_r = \{5,\ 10,\ 15,\ 20,\ 25\}$. After twelve jobs (the sixth of QC_2) the schedules start to deviate. Apparently, when $N_r = 5$ is used, the schedule is faster than all the other implemented N_r. The derived schedules when using $N_r = 10$ and $N_r = 15$ are exactly the same and perform worse than when $N_r = 5$, but better than when $N_r = \{20, 25\}$. Lastly, the worst option is using $N_p = N_r$, since it is impossible to resolve unfavorable choices at the end of computing one horizon.

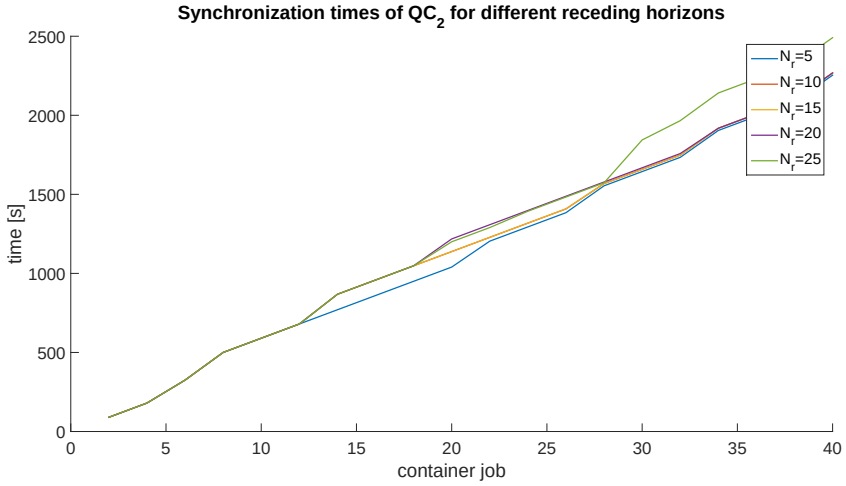

Figure 6-3: The synchronization times of QC_2 in test case 1 for the first 40 jobs.

A trade-off should be made between a high computational burden but better guarantee for an optimal schedule (small N_r) and a lower computational burden but most probably a slower schedule. Besides this consideration, the computation speed should also be taken into account. For test case 1 and $N_p = 25$ the algorithm is fast enough to calculate every five container jobs. However, this will not be the situation in test case 18.

Comparing the SMPL-MPS approach to the SMPL model

The true optimum can be found by calculating the true schedule at once (the green schedule in Figure 6-1). When a receding horizon is added, the computational burden is lowered by computing smaller subschedules. However, it has its downsides (as was pointed out in choosing N_r). For badly chosen combinations of N_p and N_r, the schedule will not remain

optimal. An example can be seen in Figure 6-4, where the first 85 container jobs are showed of the same simulation as in the previous section. From the figure it can be concluded that using a receding horizon changes the schedule, depending on the choices of receding horizon N_r and prediction horizon N_p. In the figure, the prediction horizon is 25 jobs when two QCs are performing, which is a large horizon. As already concluded; $N_r = N_p$ is not a convenient choice. Again, there occurs a delay after 25 to 30 jobs, since the optimization made an unfavorable schedule to minimize the makespan of the first 25 jobs. The algorithm is not able to correct the schedule. The other simulations perform almost the same as the optimal schedule, except for very small deviations.

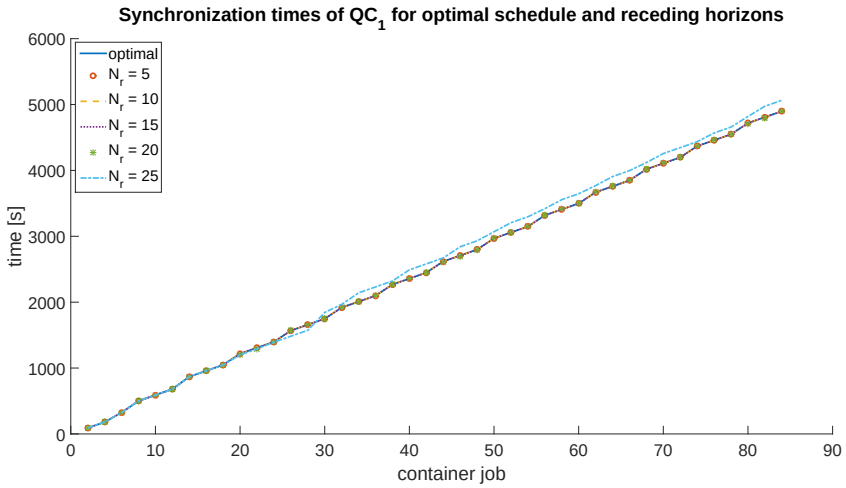

Figure 6-4: The synchronization times of QC_1 in test case 1 for the optimal schedule and the schedule that is derived using MPS.

State information in on-line scheduling

When the SMPL-MPS method is used for on-line scheduling, the information on the states can be used when a new schedule is calculated. In the beginning of this chapter, it is stated that this is one of the main advantages of using MPS, increasing the accuracy of the schedule. This is verified as follows:

- a predicted schedule is obtained using the SMPL model,
- noise is added to the AGV travel times (described in Chapter 7),
- the obtained SMPL schedule is simulated (with noisy travel times) as an LP problem, deriving the "true schedule",
- a predicted schedule is obtained using the SMPL-MPS model,
- the obtained SMPL-MPS schedule is again simulated with the same noisy times, deriving a second "true schedule".

In this work it is assumed that on the moment of rescheduling (say $\boldsymbol{\kappa} + \iota$) all information on previous states are available, due to time limitations. It should be noted that this is a wrong

assumption, but the idea of using state information still applies when less data is available.

The verification was done for test cases 1 ($N_p = 50$), 8 and 17 ($N_p = 25$). In these simulations many different paths can be chosen in the system due to either many vehicles or a large prediction horizon. In test cases 8 and 17, the large amount of previous cycles P implies that a considerable amount of state information can be used. In test case 1, $P = 5$ and therefore it was necessary to use a larger distribution for the noise in the real data to make the differences noticeable. Four schedules for each test case are obtained and presented in Table 6-2 with the sum and maximum of all QC times. The table supports the claim that using state information increases the accuracy of the prediction, since the SMPL-MPS prediction is closer to the real values than the SMPL prediction. The true schedule is faster for the SMPL-MPS model than for the SMPL prediction.

test case 1

	SMPL	true schedule	SMPL-MPS	true schedule
QC_1	3148	4280	3877	4140
QC_2	3153	4308	3876	4097
sum	6301	8588	7753	8237
max	3153	4308	3877	4140

test case 8

	SMPL	true schedule	SMPL-MPS	true schedule
QC_1	901	1018	940	1018
QC_2	919	997	838	922
QC_3	863	929	874	933
QC_4	829	945	830	945
QC_5	810	914	956	959
sum	4322	4803	4438	4777
max	919	1018	956	1018

test case 17

	SMPL	true schedule	SMPL-MPS	true schedule
QC_1	630	665	673	703
QC_2	829	849	739	747
QC_3	633	668	770	778
QC_4	759	791	648	675
QC_5	795	837	701	712
QC_6	673	747	824	846
QC_7	796	827	805	838
QC_8	821	869	814	834
QC_9	839	853	832	853
QC_{10}	668	793	843	850
sum	7443	7899	7649	7836
max	839	869	843	853

Table 6-2: The QC synchronization times of test cases 1, 8 and 17 for the SMPL- and SMPL-MPS prediction model and their true schedules.

Chapter 7

Stochastic Switching Max-plus-linear Model

In the SMPL model, some parameters in the system are assumed to be constant. However, there is a lot of perturbation in this system. Lets consider the variable parameters as defined in the SMPL model:

- τ_v: the AGV travel times between cranes,
- τ_2: the QC container handling times,
- τ_6: the YC container handling times.

The aim is to consider stochastic perturbed AGV travel times. Therefore, these will be noisy in the SSMPL model. Varying QC times will also be considered. The YC handling times are still assumed to be deterministic.

In the SMPL system, both real-valued variables and binary variables are present. The optimal schedule can be obtained when a MILP problem is formulated and optimized. However, for the SSMPL system, the optimization curve in the MILP problem will become a polynomial of high order and therefore a numerical integration would be time-consuming. Deriving an analytical equation that describes this polynomial will again be very time consuming, as will be the calculation of the optimum due to the complexity of the system. Therefore, a proper alternative would be to settle for a near-optimal solution. When "just" a near-optimal solution is desired, there are multiple manners to solve the problem, e.g. using an approximation method for computing the expected value of max-plus systems [44], an ordinal optimization or randomized algorithms such as the Monte Carlo algorithm. These algorithms do no longer require the exact polynomial curve of the MILP framework.

7-1 Monte Carlo method

In this thesis, a Monte Carlo algorithm is implemented to avoid the analytical burden of calculating the expectation of the performance index by obtaining an empirical mean. For this, a certain number of realizations should be created to make sure the computation is efficient. The stochastic disturbances in these multiple realizations are chosen randomly, according to the distribution of the stochastic variables [45]. After the determination of the number of realizations, the optimal schedule is determined over the R number of realizations.

Assume random variable set Θ and a probability density measure P on Θ. J is measurable with respect to the probability function P. The expectation of the performance index J can now be expressed as:

$$\mathbb{E}J = \int_{\Theta} J(\theta) dP. \tag{7-1}$$

The Monte Carlo algorithm approximates Eq. (7-1) by using the empirical mean:

$$\hat{\mathbb{E}}J := \frac{1}{R} \sum_{r=1}^{R} J(\theta_r), \tag{7-2}$$

where all θ_r are independent, identical distributed (i.i.d.) samples from Θ: $\vartheta = \{\theta_1, ..., \theta_r\}$. In words, Eq. (7-2) means that the empirical mean of the cost function is stated to be equal to the average of R realizations, were every realization has a unique set of distribution variables θ_r.

Of course, if $R = 1$ is chosen, there is no guarantee that the empirical mean is close to the expected value. If $R \to \infty$ the empirical mean $\hat{\mathbb{E}}J$ converges in probability to the "real" $\mathbb{E}J$. This can be seen when the error is introduced. Since stochastic systems are considered, the error is expressed in a confidence interval:

$$\text{Prob}\left(|\hat{\mathbb{E}}J - \mathbb{E}J| < \epsilon\right) \geq 1 - \frac{(Var\ J)^2}{R\epsilon^2}, \tag{7-3}$$

where ϵ is the error and Var J the variance of J. Now if $R \to \infty$, then for all $\epsilon > 0$.

$$\text{Prob}\left(|\hat{\mathbb{E}}J - \mathbb{E}J| < \epsilon\right) \to 1. \tag{7-4}$$

Thus the probability that the absolute difference between the empirical mean and the expected value is smaller than ϵ, is (almost) one. Commonly used confidence intervals are 95% or 99%. Suppose 95% reliability of the estimation procedure is desired and $\epsilon = 0.05$ is chosen, then:

$$0.95 \geq 1 - \frac{(Var\ J)^2}{0.05R} \tag{7-5}$$

When both the confidence interval and error are chosen and additionally the variance of the performance index is calculated, the number of samples R can be obtained. The variance is not calculated analytically for the same reasons as the real expected value is not calculated. Therefore, it needs to be determined experimentally. Next, an explanation of the distribution of the variables is given.

7-1-1 Weibull distribution

Nor in literature, nor common knowledge provides a specific distribution for the container handling times in a terminal. It is however possible to obtain a likely distribution by reasoning. When there is a lack of accurate information about the distribution of a variable, the Gaussian or uniform distribution are widely accepted. It is however argued that the Weibull distribution fits the problem better. This distribution is more often used in scheduling problems [7, 32].

The QC will never exceed its optimal performance. The added time here is zero: e_0. On average, it will perform slightly above its optimal performance, say $e_{av/h}$ seconds. Suppose that in the worst case, the QC takes twice the optimal handling time (adding e_{wc} to the optimal handling time). The uniform distribution would not be very accurate. It is highly doubtable that the optimal, average and worst performance occur the same rate. Furthermore, the average performance is not the average of "optimal performance + worst performance", such as a Gaussian distribution implies. Moreover, the distribution is asymmetric and it should be skew with the mean closer to the optimal performance (positive skew). A convenient choice for a distribution is that of Weibull, since it fits the description of the desired shape and is also used in similar types of problems, such as train scheduling [46, 47]. An example of the distribution can be seen in Figure 7-1. Note that the average performance per hour is the mean. The blue area and green area under the curve are equal.

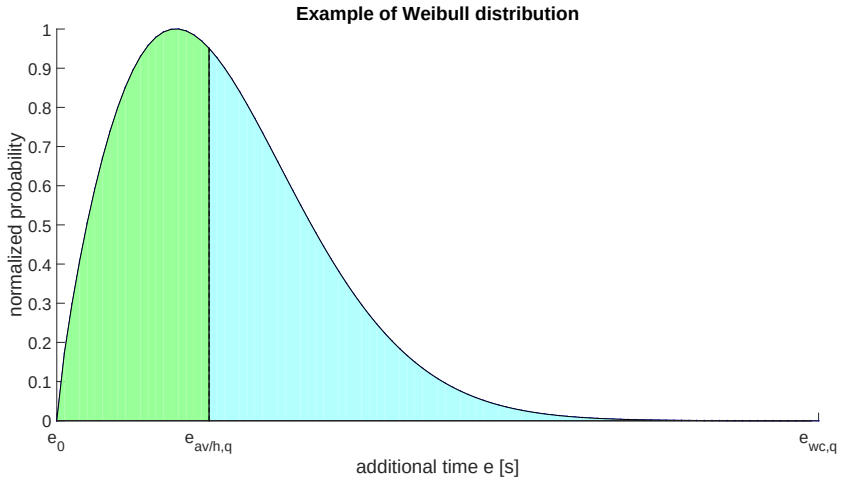

Figure 7-1: Example of a Weibull distribution with the additive noise e, divided in optimal performance e_0, average performance loss per hour $e_{av/h,q}$ and worst performance $e_{wc,q}$.

The same reasoning can be used for the AGV travel time, only now the time that is lost during the journey should depend on the distance the AGV travels. Since a linear relation between the AGV travel time and distance will be assumed, the lost time depends linearly on the AGV travel time. Now $e_{av/h}$ and e_{wc} in Figure 7-1 is not a hard number, but for example respectively 20% and 200% of the travel time ($0.2T$ and $2T$).

The probability density function of a Weibull distribution is described as:

$$f(x;\eta_{wb},\rho_{wb}) = \begin{cases} \dfrac{\rho_{wb}}{\eta_{wb}} \left(\dfrac{x}{\eta_{wb}}\right)^{\rho_{wb}-1} e^{\left(-x/\eta_{wb}\right)^k} & x \geq 0 \\ 0 & x < 0. \end{cases} \tag{7-6}$$

Here, $\rho_{wb} \in (0,\infty)$ is the shape parameter and $\eta_{wb} \in (0,\infty)$ the scale parameter. The former determines the shape of the distribution, the latter determines how wide the distribution spreads. A number of combinations will be evaluated, all satisfying the defined e_0, $e_{av/h}$ and e_{wc}.

7-1-2 Performance index

The only objective that will be considered is that of minimizing the makespan. The YC optimization has a slightly worse performance and the AGV travel distance has a very unfavorable effect on the makespan. Therefore, these will be disregarded. To determine the objective of the optimization of the SSMPL model is then straightforward. First of all, an optimal average performance is desired. Minimizing this component of the cost function aims for a minimum makespan *most* of the time. Due to the stochastic nature of the model, it however might happen that the performance in few cases is far from optimal. Even when the probability of such an event is very low, it would still be disastrous for this system. Therefore, a second, weighted cost is added to the performance index: the weighted worst case scenario. The minimization of the performance index will be formulated as:

$$\min_{t,x,z} J = \min_{t,x,z} \left(\frac{1}{R} \sum_{r=1}^{R} t_r + \lambda_r \max_r t_r \right) \quad \text{for } r = 1,...,R. \tag{7-7}$$

Here λ_r is the weighting factor. If λ_r is chosen much larger than one, the minimization of the performance index will focus on minimizing the worst case. If $\lambda_r = 1$, the trade-off between the average performance and worst case scenario is equal. Logically, when $0 < \lambda_r < 1$, the average performance has more weight in the minimization of the performance index. Lastly, if $\lambda_r = 0$, minimizing J equals the minimization of the average performance.

The parameters in the cost function:

- R: number of realizations
- t_r: maximum QC synchronization time of the last cycle of one stochastic realization:

$$t_r = \max_i x_i(k+N_p) \quad \text{for} \quad i \in (1,...,N) \tag{7-8}$$

Earlier it was already explained that all cycles $k+j$ in a realization should have some weight in the implemented optimization. This is again neglected in the formulation of the performance index. To clarify the expressions above, a small example is given.

Example 7-5. *Suppose the number of realizations R equals 2. The weight λ_r is set to* 0.5. *There are two QCs and one YC, thus $x(k) \in \mathbb{R}^{3\times 1}$. Then*

$$\min_{t,x,z} J = \min_{t,x,z} \left(\frac{1}{2}\left(t_1 + t_2\right) + 0.5 \max\left(t_1, t_2\right) \right)$$

Suppose in respectively the first and second realization, the states regarding the QCs become:

$$\begin{aligned} \textit{for } r = 1: \quad x_{r=1}(k+N) &= \begin{bmatrix} x_{1,r=1}(k+N) \\ x_{2,r=1}(k+N) \end{bmatrix} = \begin{bmatrix} 1150 \\ 1390 \end{bmatrix} \\ \textit{for } r = 2: \quad x_{r=2}(k+N) &= \begin{bmatrix} x_{1,r=2}(k+N) \\ x_{2,r=2}(k+N) \end{bmatrix} = \begin{bmatrix} 1130 \\ 1340 \end{bmatrix}. \end{aligned}$$

Then t_1 and t_2 are calculated as follows:

$$\begin{aligned} t_1 &\geq x_{1,r=1}(k+N) \\ t_1 &\geq x_{2,r=1}(k+N) \\ t_2 &\geq x_{1,r=2}(k+N) \\ t_2 &\geq x_{2,r=2}(k+N) \end{aligned}$$

thus

$$\begin{aligned} t_1 &\geq 1390 \\ t_2 &\geq 1340 \end{aligned}$$

If the variable t_{wc} is introduced for the worst case, it is obtained that:

$$\begin{aligned} t_{wc} &\geq t_1 \\ t_{wc} &\geq t_2 \end{aligned}$$

thus

$$t_{wc} \geq 1390$$

The minimization of the performance index then becomes:

$$\begin{aligned} J_{min} &= \frac{1}{2}\left(1390 + 1340\right) + 0.5 \times 1390 \\ &= 1365 + 695 = 2060 \end{aligned}$$

△

7-2 Results of the SSMPL model

The distribution functions will be defined and the number of realizations necessary to approximate the mean of the expected value is determined. This will be a trade-off between accuracy, problem size and solution times. Lastly, the SMPL and SSMPL models are compared.

Distribution function

To determine the parameters of Eq. (7-6), the shape of the Weibull distribution is chosen by analyzing a number of plots and set to be $\rho_{wb} = 2$. The second parameter in Eq. (7-6) can be fit such that the mean of the distribution is about 20s for the QC ($\eta_{wb} = 23$) and $0.2 \times T$ for the AGV. The noise e can be added by using the MATLAB function *wblinv*. This function contains an analytically expression of the inverse of the cumulative distribution function (CDF) and selects the noise related to the given (random) points between 0 and 1, as is clarified in Figure 7-2. Note that the chance of $50 \leq e \leq 80$ is very slim, while it is very well possible that $15 \leq e \leq 25$.

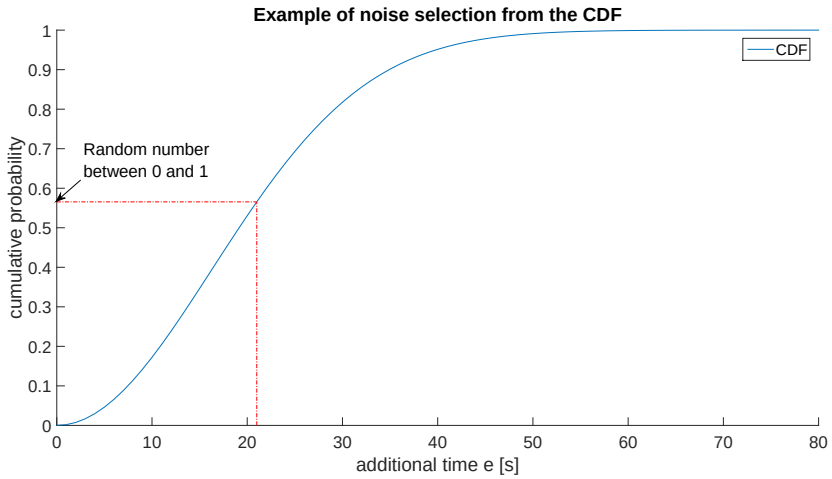

Figure 7-2: The noise is chosen by picking a random number between 0 and 1, here 0.5655. The additional time e corresponds to the projection on the x axis following the chosen distribution.

Number of realizations

Now that the distributions are defined, the number of realizations can be determined. Of course, more realizations results in a better estimate (see Eq. (7-5)), but choosing a large number of realizations results in memory problems and long computation times. The maximum number of realizations for the largest test case equals 12. The Monte Carlo algorithm increases the number of constraints by approximately:

$$nr_{constr}(SSMPL) \approx R \times nr_{constr}(SMPL). \tag{7-9}$$

Therefore, some simple cases are defined to be able to make a large number of realizations. The alternative cases are shown in Table 7-1, where also handling time adjustments are done.

In Figure 7-3 it can be seen that the number of realizations influences the accuracy of the estimation. When 250 realizations are used, the makespan is much more reliable than when only 5 realizations are used. However, due to the memory restrictions, the number of realizations in the Monte Carlo algorithm is chosen to be 10 (Figure 7-3).

test alternative	QCs	YCs	AGVs	P	N_p
1	1	1	1	2	5
2	1	2	2	5	5
3	2	4	4	8	5

Table 7-1: The newly defined alternative test cases to simulate with large number of realizations.

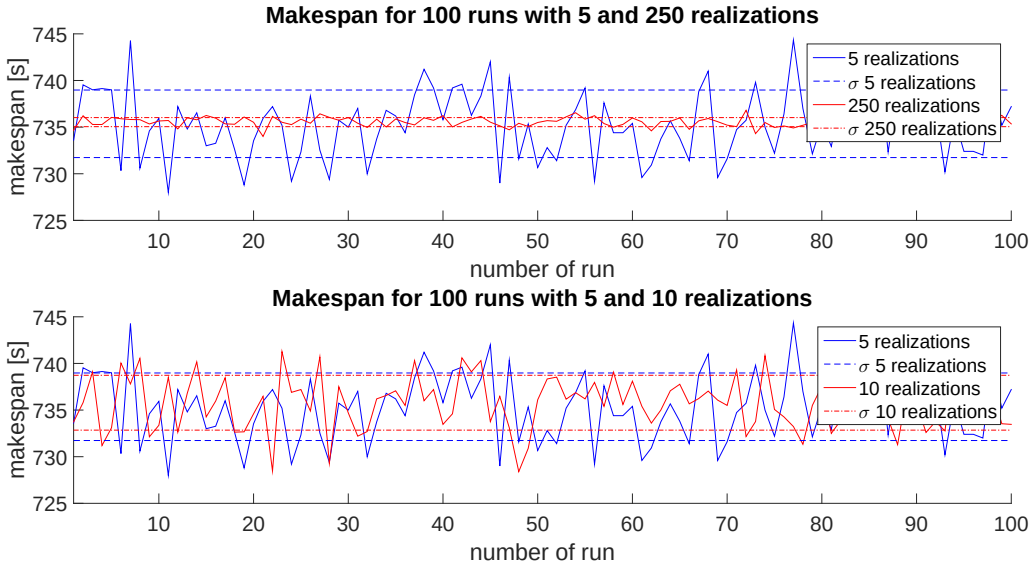

Figure 7-3: A typical representation of the makespan in 100 simulations when 5, 10 or 250 realizations are used (alternative test case 2).

In Figure 7-4 the development of the standard deviation can be seen, which is empirical determined. To check how valid one simulation with a particular number of realizations is,

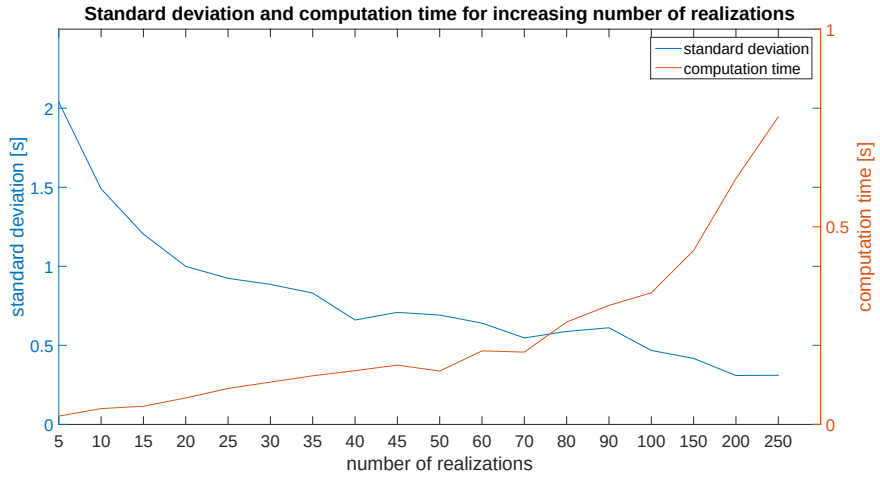

Figure 7-4: The empirical standard deviation of the makespan decreases with the number of realizations for all alternative test cases (here case 2).

100 simulations are done for this certain R (R is the number of realizations). When R is low, it is expected that the mean of all t_r (as defined in Section 7-1-2) has a larger deviation than when R is high. Low R causes large fluctuation in the first part of the performance index (the mean of t_r) in Eq. (7-7), resulting in a less reliable estimation. When the number of realizations in increased, more samples are taken from the distribution and the estimation is less dependent on unfortunate distribution choices for one single realization. The trend in Figure 7-4 is therefore expected. The figure also shows the increase in computation time. For this small test case, the runtime of the solver is still below one second, but again the trend is typical for larger test cases.

SSMPL model results

First, the influence of the distribution of the QC and AGV handling times is tested by using different Weibull parameters. The performance index is analyzed and the results of the SMPL model are compared to those of the SSMPL system.

The influence of the amount of noise on the QC and AGV are tested by increasing the expected time loss up to respectively 20s and 0.2% per move. With the shape parameter $\rho_{wb} = 2$, the mean is 7, 14 and 20s for the QC and 0.1% and 0.2% for the AGV.

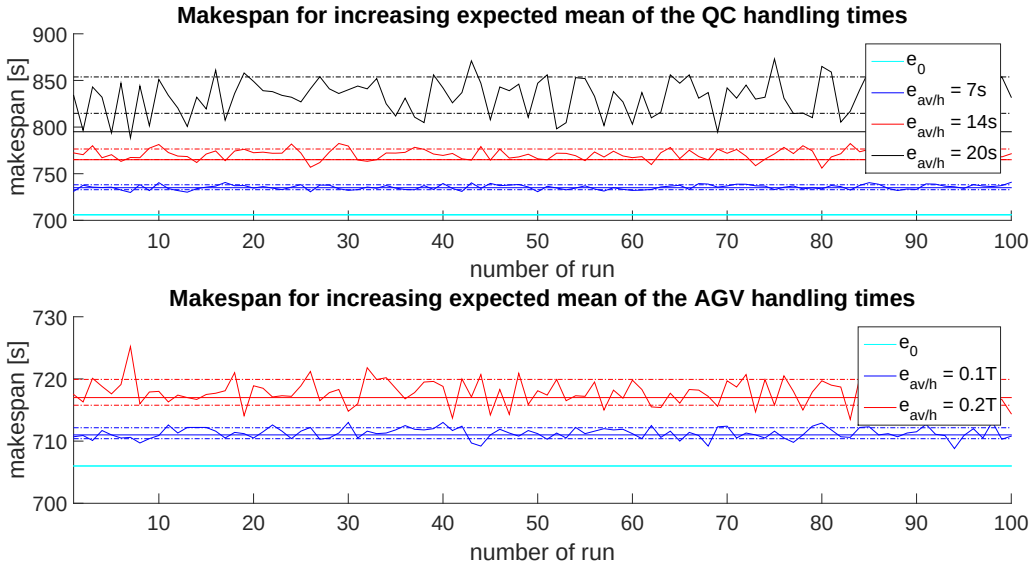

Figure 7-5: Makespan for increasing expected mean of both QC (top) and AGV (bottom) handling times. The dotted lines represent the standard deviation in the 100 simulations, while the horizontal lines represent the nominal values.

As the expected mean of the distribution increases, the makespan also increases. Secondly, the makespan objective is more scattered when the stochastic variables are taken from a wider distribution. Another observation is that the makespan is more often larger than the nominal makespan (with purely the expected means). This is due to the max-plus algebra, were

synchronization can only occur when two preceding events are finished. For the preceding handling times, only one of the following four situations results in a faster schedule:

- the crane handling time slower + the AGV travel time slower → synchronization later,
- the crane handling time slower + the AGV travel time faster → synchronization later,
- the crane handling time faster + the AGV travel time slower → synchronization later,
- the crane handling time faster + the AGV travel time faster → synchronization earlier.

A last important remark is that the schedule is always less tight than that of the optimal performance. Applying multiple realizations with distributed variables will most probably result in a more robust schedule. When the variables are deterministic, the algorithm can choose an AGV that is exactly on time. When the variables are distributed, the algorithm "realizes" that this decision might result in a delay and can handle accordingly.

Performance index

The performance index is discussed in Section 7-1-2 and is a trade-off between optimizing the average performance of all realizations and limiting the worst case. In the container terminal system the optimization of solely the worst case will not be convenient. The trade-offs that are investigated are $\lambda_r = \{0,\ \frac{1}{R},\ 0.5,\ 1,\ 10\}$. The same set of stochastic variables is used in the optimization to compare the λ_r-s fairly.

In test case 1, the results for $\lambda_r = \{0,\ \frac{1}{R}\}$ are equal and the resulting schedules of $\lambda_r = \{0.5,\ 1,\ 10\}$ are also the same. In Figure 7-6 the difference is displayed when the worst case scenario is taking into consideration (black line) or when it is (almost) disregarded (blue line). The gain in reducing the absolute maximum is large when the worst case is weighted in the performance index, while it only slightly compensates on the average maximum. In the lower plot in Figure 7-6 it shows that minimizing the worst case also results in a slight decrease in average QC synchronization times. This might be a coincidence, since this criteria is not included in the performance index. Figure 7-6 is typical when multiple sets of distributed variables for test case 1 are analyzed.

The same analysis is performed for test cases 8 and 17. In test case 8, no difference appears when the objective function is adjusted. This can be explained by the ratio of AGVs to QCs, which is high. Therefore, the AGV cycles are not critical and the makespan can not be influenced by varying weights in the objective function.

In test case 17 there is no difference in the maximum synchronization times. The choices in the optimization are "too obvious" to the Gurobi solver, since the relative high number of AGVs ($V = 45$) to the schedule container jobs ($N_p = 15$). Interesting to see is that, while the maximums remain equal, the average QC synchronization times differ. This is possible, because the sum of the last QC synchronizations is not specified in the objective function. In Appendix K, Figure K-1 shows the average QC synchronizations.

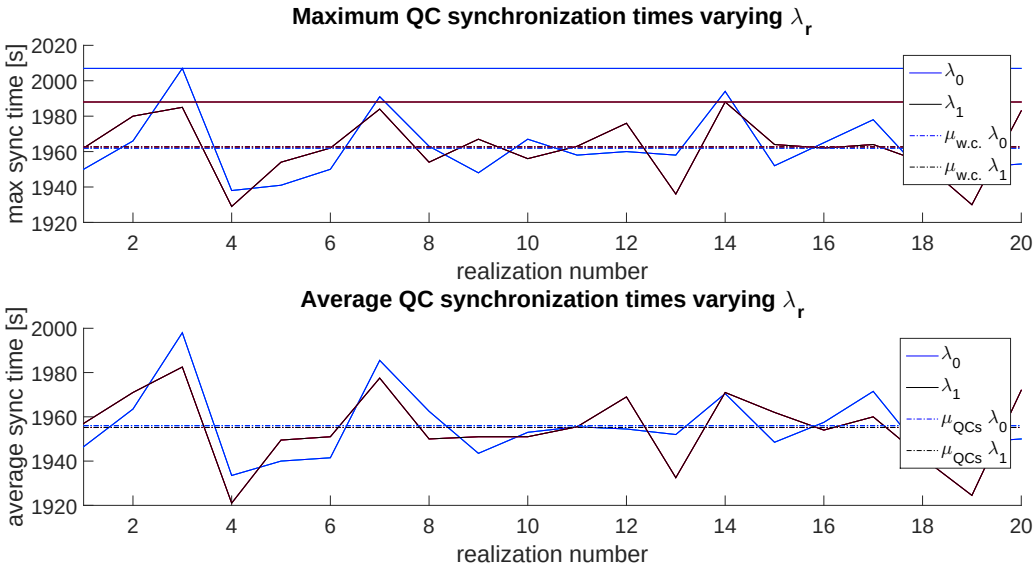

Figure 7-6: The maximum and average QC synchronization times in test case 1, when the performance weight λ_r is varied. $\lambda_0 = \{0,\ \frac{1}{R}\}$ and $\lambda_1 = \{0.5,\ 1,\ 10\}$.

Validation of the model

The SSMPL model is presented in this chapter. To estimate the value of the stochastic approach, the difference is discussed by calculating schedules using three different models and verifying their results in a number of "real-case" simulations. The models that are simulated are:

- the SMPL approach assuming optimal performance,
- the SMPL approach assuming nominal performance,
- the SSMPL approach using stochastically distributed variables (Monte Carlo approach).

Since noise and modeling errors by assumption are multiplicative in the max-plus algebra, the SSMPL system results in a MILP problem of the form:

$$\operatorname{diag}(\mathbf{E}_1, ..., \mathbf{E}_R) \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_R \end{bmatrix} + \beta \begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_R \end{bmatrix} \mathbf{z} \leq \operatorname{diag}(\mathbf{B}_1, ..., \mathbf{B}_R) \begin{bmatrix} \boldsymbol{\Theta}_1 \\ \vdots \\ \boldsymbol{\Theta}_R \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_R \end{bmatrix} + \beta \begin{bmatrix} \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_R \end{bmatrix} \mathbf{w} \quad (7\text{-}10)$$

where only the AGV travel times are perturbed.

Initially, test cases 1, 8 and 17 are tested, using the following system properties:

- $\lambda_{max} = 1,\ \lambda_{sum} = 0$ in the performance index of Eq. (5-13),
- only the AGV travel times are perturbed by noise,

- this noise is distributed as described on Page 68,
- $\lambda_r = \frac{1}{R}$ in the performance index of Eq. (7-7),
- the number of realizations for the Monte Carlo algorithm is 20 (test case 1), 15 (test case 8) or 10 (test case 17),
- the prediction horizon is $N_p = 25$ (test cases 1 and 8) or $N_p = 15$ (test case 17).

In Chapter 5 it is argued that the performance index Eq. (5-13) should be a trade-off ($\lambda_{sum} > 0$). This is however not realized, due to the formulation of the variable t_r in Eq. (7-8).

When the three derived schedules are simulated in 1000 real cases, it immediately becomes clear they are exactly equal in test case 8. A plausible explanation is that the AGV cycles are not critical. In Chapter 5 it is argued that the QC waiting times are resolved when $V = 3N$-$4N$. In the test case, $V = 4.2N$ *and* the AGV travel times are relatively short compared to those of e.g. test case 17. By reducing the AGV speed to 1.5m/s, it is ensured that its cycles become critical. A second observation is that it is not possible to simulate sufficient container jobs to notice differences in QC_1 to QC_5 for test case 17.

The actual performance should be measured by the average maximum QC synchronization times, due to the formulation of the performance index. For both test cases 1 and 8, the SSMPL approach resulted in the lowest times. The standard deviations are relatively low in test case 8. It is possible that the AGV cycle is often still not critical or too few choices are in the optimization. When using nominal values, a better performance is obtained than when optimal values are assumed. Based on the three test cases (test case 17 showed the same trends as can be seen in Appendix K) it is concluded that using distributed values indeed enhances the performance. This is probably due to the avoidance of situations where the AGV is just in time to pick up a container at a crane.

The question arises if the Monte Carlo algorithm is (too) conservative in avoiding these "just in time" situations, especially since the number of realizations is low for test case 17. To influence the choices made by the algorithm, the variables are adjusted such that they are less distributed. A probability density function (PDF) is introduced ($\rho_{wb} = 20$ and corresponding η_{wb}) that has a sharp peak at 20% travel time. The results can be seen in Appendix K, Table K-1. Since the adjustment did not result in considerable enhanced performance, the number of realizations is sufficient and the original approach does not need adjustment.

Lastly, it should be noted that the performance on the *sum* of the QCs in the SSMPL approach might be worse than that of when nominal values are assumed (Table 7-2). This is related to the objective function.

7-3 MPS for the SSMPL model

The previous explained advantages of introducing a receding horizon still applies to the system. When this schedule is used on-line, state information can again enhance the performance

test case 1		Optimal	Nominal	SSMPL
QC_1	mean	1960	1961	1958
	std	25	26	22
QC_2	mean	1960	1954	1951
	std	24	24	20
sum	mean	3920	3915	1909
	std	48	48	41
max	mean	1964	1963	1959
	std	24	25	21

test case 8		Optimal	Nominal	SSMPL
QC_1	mean	1206	1201	1200
	std	6	3	0
QC_2	mean	1081	1085	1080
	std	3	7	0
QC_3	mean	1109	1080	1109
	std	11	1	13
QC_4	mean	1098	1081	1080
	std	15	5	0
QC_5	mean	1101	1083	1081
	std	18	8	3
sum	mean	5594	5530	5550
	std	27	12	13
max	mean	1206	1201	1200
	std	6	3	0

Table 7-2: The average times and standard deviations for the QC synchronizations in test cases 1 and 8. The average summation and maximum of all 1000 runs with their standard deviations are also shown.

and the SSMPL-MPS method is obtained. Important to note is that this method has slight differences with respect to the SMPL-MPS approach. When a job enters in the horizon, new noisy travel times are introduced. For jobs that are already defined in the previous horizon κ, the handling times remain equal. The on-line and off-line approach need different implementations:

- Off-line: The SSMPL optimization uses R realizations, which all have unique states $\mathbf{x}_r$, as shown in Eq. (7-10). These states will be used as initial condition for the next optimization, when the horizon shifts.
- On-line: The true state is calculated using an LP optimization, which serves as the initial condition in the rescheduling.

Earlier, it is argued that it is not convenient to schedule far ahead in time, due to the perturbations in the container terminal system. Now that these uncertainties are introduced in the model, great deviations at the end of schedules for a large amount of jobs are to be expected for the off-line scheduling.

7-4 Results of the SSMPL-MPS approach

As seen in Eq. (7-9) the problem size of the Monte Carlo algorithm is even larger than the SMPL approach. This restricts the prediction horizon and number of realizations to e.g. $N_p = 15$ and $R = 10$ for test case 17. In the following simulations, test case 1 is used. It allows more realizations, larger prediction horizon and showed the best results in Section 7-2. Based on these results, $\lambda_r = 0.5$. Additionally, the prediction horizon is chosen as $N_p = 25$, the number of realizations $R = 20$ and the rescheduling horizon as $N_r = 15$ (considering the trade-offs made in Section 6-2).

Receding horizon

State information in on-line scheduling

Chapter 8

Conclusions & Recommendations

In this final chapter, the SMPL, SMPL-MPS, SSMPL and SSMPL-MPS models are shortly discussed and the most important conclusions from this work are drawn. Finally, recommendations are done for future research.

8-1 Conclusions

From the results of the SMPL-, SMPL-MPS, SSMPL- and SSMPL-MPS approaches, conclusions are drawn, suggestions are made and design rules are formulated.

Most of the MILP inequality constraints are derived by obtaining the model equations using graph theory and by the parameterization of the introduced max-plus binary control variables. This proved to be an effiecent method. However, due to the formulation of the model and max-plus binary control variables, small problems already result in enormous matrices. This often causes memory problems. As long as the model is derived as explained in Section 4-1 and it is desired to describe all possible path in the system, this problem remains and is a drawback of this method.

8-1-1 SMPL model

An optimal schedule for the SMPL model is calculated using the MILP- and Benders' decomposition approaches, both discussed next.

MILP approach

General design rules can be formulated from the analysis of the parameters in the MILP formulation.

- The Gurobi and CPLEX solvers perform best *for a MATLAB implementation*, not guaranteeing the same result for other programming languages. A disadvantage is that they require a license, which might be costly to the user. The CBC solver is the best open-source solver, but does not perform satisfiable. The solvers highly influence the solution times.

- The best performing **objective function** is the optimization of the QCs. A trade-off ($\lambda_{sum} = 1$ and $0 < \lambda_{max} > 0.5$) between the sum and the maximum of last QC synchronizations is recommended (Table 5-6). The value of λ_{max} should not be chosen too large, since an overall optimal performance is desired when a receding horizon is added to the SMPL model.

- In this work, the **prediction horizon** depends on computation time, scheduled horizon in time and the capacity of hardware to solve the problems. This is explained in Section 5-3-2 and shown in Table 5-8 and Table 5-9. If it is assumed the hardware is sufficiently advanced to solve the schedule without being limited by the computation times or memory capacity, the most important criteria to choose the prediction horizon is the scheduled horizon in time. This depends on the the accuracy of the prediction and the cycle time and number of QCs. A larger prediction horizon might be used when a high accuracy in prediction is obtained, when e.g. a terminal is not busy and AGVs are seldom delayed. If the QC cycle time is low or the number of QCs high, the prediction horizon increases with respect to when the QC cycle time is high or the number of QCs is low. This due to the number of container jobs that occur in the same amount of time.

- The **number of AGVs** influence the dynamics of the system enormously. If too little AGVs are available, the QCs have large waiting times (Figure 5-9), which is very undesirable. To anticipate on the investment in unnecessary capital, too many AGVs should also be prevented. If the system is deterministic, an estimation of the number of AGVs can be made by (assuming sufficient amount of YCs):

$$\frac{cycle\ time\ QC}{N} \approx \frac{cycle\ time\ AGV}{V} \tag{8-1}$$

  where N the number of QCs and V the number of AGVs. The left part of the equation represents the average time interval of which a container job is ready at the quay. Choices of the cycle time could be the average or maximum (utterly conservative) cycle time. This can be verified using the results of the simulations (Figure 5-9). When the average AGV cycle time is used (including a penalty for delays), the optimal number of AGVs can be found as follows:

$$\begin{aligned} cycle\ time\ AGV &= (average\ QC - YC\ travel\ time) * 2 + \tau_3 + \tau_5 \\ \frac{120}{3} &\approx \frac{461}{V} \\ V &\approx 11,5 \\ V &\approx 3,8N \end{aligned}$$

  as Figure 5-9 shows. In conclusion, the number of AGVs necessary to reduce QC waiting times relates to their own cycle time, the number of QCs and their cycle times.

- For the **number of previous cycles**, $P = 1.6V$ is recommended as setting. Larger P will not enhance performance, but will increase the computational burden. When $P < 1.6$ is chosen, the optimization loses too many freedom in its scheduling, as is explained in Section 5-3-2.

A last important remark is that it is highly recommended to use sparse matrices in the implementation of (all the) SMPL model(s), reducing the memory usage.

Benders' decomposition approach

The results of the Benders' decomposition approach are disappointing. When this method is compared to the MILP approach, it performs much worse. The Benders' decomposition approach reaches the same optima in simulations as the MILP approach, but uses much longer computation times (Figure 5-12). Alternative Benders' cuts are proposed, but it is concluded that they do not solve the problem that is adduced.

8-1-2 MPS for SMPL model

Using MPS proved to be sufficient for two main reasons:

- since it will not be accurate to schedule hours ahead, a smaller prediction horizon can be adapted, reducing the computation times,
- new state information can be taken into consideration when a new horizon is scheduled, increasing the accuracy.

The computation speed might reduce approximately six times (Figure 6-2). The possibility that an optimization continues for longer time than expected should always be taken into account. The newly introduced scheduling parameter N_r (discussed in Section 6-2) and the already defined parameters λ_{sum}, λ_{max} and N_p are considered. Lastly, conclusions are drawn from the off-line and on-line validation of the SMPL-MPS approach.

- The same conclusions as in the SMPL model can be drawn for the **performance index**.
- In the MPS approach, **rescheduling horizon** N_r is closely related to **prediction horizon** N_p, where N_p should be chosen as explained in Section 8-1-1. The rescheduling horizon should not be taken too small, since this will result in higher computational burden without increasing the performance. Choosing a large ratio N_r/N_p will result in a performance drop, due to unfavorable choices made at the end of one (horizon) schedule. The next optimization should be able to correct for these choices by shifting the horizon before it is reached. From simulations it is obtained that $N_r/N_p = 4/5$ can still result in delays, while $N_r/N_p = 2/3$ performs approximately equal to the SMPL approach. It is therefore advised to take the ratio $1/3 \leq N_r/N_p \leq 2/3$.

- In the **off-line validation**, there is no state information available. It should be verified that the SMPL-MPS approach calculates an optimum that is approximately equal to the optimum of the SMPL model. With the correct N_r/N_p ratio this appears to be the case. As is concluded, the MPS implementation usually reduces the simulation time, but using this validation it is also assured that the optimality of the schedule is not lost.

- In the **on-line validation**, new state information is available. Using this state information is of great advantage. In test case 1, approximately one hour is scheduled, where the on-line SMPL-MPS model reduces the makespan approximately five minutes compared to the off-line method. Increasing the accuracy of the prediction is valuable to this model and measurements on the state are highly recommended.

8-1-3 SSMPL model

In the SSMPL approach an optimal schedule is derived, using a Monte Carlo algorithm which introduces (Weibull) distributed variables for the AGV travel times. The Monte Carlo algorithm uses multiple realizations (changing the performance index) to avoid choices in which the AGV is just in time. An important disadvantage of the algorithm is the increasing problem size by the number of realizations. From the analysis in Section 7-2, numerous conclusions can be drawn.

- The **Weibull distribution** is recommended, since it has matching properties with the expectation of the distribution of the variables, as argued in Section 7-1. The Weibull parameters are an indication for the shape of the distribution function. In this work, the average AGV travel time is 20% above the optimal travel time. These values are also an indication and container terminal specific.

- A very important consideration is that of choosing **the number of realizations**. More realizations provide a better estimation of the performance index. A proper design tool is relating the number of realizations to the variance of the performance index, using:

$$\text{Prob}\left(|\hat{\mathbb{E}}J - \mathbb{E}J| < \epsilon\right) \geq 1 - \frac{(Var\ J)^2}{R\epsilon^2}, \tag{7-3}$$

  and shown in Figure 7-4. General accepted values are $\epsilon = 0.05$ and a 95 or 99% confidence interval. The variance of the performance index should be determined empirically. Due to memory issues in this work, it is not possible to use more than 10 realizations for large terminals, which is relatively little.

- In the **performance index**, a trade-off is made between minimizing the worst case scenario and optimizing the overall performance, as Eq. (7-7) shows. It is recommended to use $0.5 < \lambda < 1$. The average performance is the primal objective, since an overall optimal performance is desired. However, the worst case scenario in the container terminal system would be disastrous and should have weight in the minimization.

- By **validating the SSMPL model** it becomes clear that the SSMPL approach has the most optimal performance compared to the SMPL model (nominal and optimal AGV

travel times assumed). The addition of distributed variables is highly recommendable, since it showed this positive influence on the schedule robustness when AGVs are the critical cycle.

8-1-4 MPS for SSMPL model

Als dit nut heeft om nog naar te kijken

8-1-5 Final conclusion

In the beginning of this thesis, the following research question is formulated:

What are the current possibilities and future perspectives for scheduling AGVs in an automated container terminal using max-plus-linear systems?

Two important criteria are emphasized: the approach should be fast, such that it can be used on-line, and it should be robust, such that it considers the uncertainties in the system. The research question is answered by exploring several methods that take the two criteria into account; the SMPL-, SMPL-MPS-, SSMPL- and SSMPL-MPS approaches. The SMPL-MPS approach provides schedules within seconds for large container terminals, satisfying the speed-criteria. The SSMPL system appears to be more robust, but takes longer to compute. Since the SSMPL performed better in the validation, it is recommended to implement distributed variables.

In conclusion, the possibilities are explored and show promising results, but are not refined sufficiently to use in the industry. A recurrent problem is the memory capacity, which can be resolved by using more advanced hardware. With the ongoing developments in hardware and (MILP) optimization algorithms, this issues will probably be solved in the near future. Lastly, assumptions are made on the exact dynamics and the distribution in the container terminal. Several resources provide different crane handling times, e.g. the company TBA, Duinkerken et al. (2000) [18] or Steenken et al. (2004) [3].

8-2 Recommendations for future research

Recommendations are made for future research, of which first the most important ones are presented.

- The importance of job order switching is explained, but is not implemented due to time restrictions. Major gaps in QC synchronization times occur when the cycle type is 2: "SYYS". A solution is suggested and further explained in Appendix H, which should resolve the delays.

- In the rescheduling of the (S)SMPL-MPS schedules, the assumption that all state information is available is false. The implementation should be adjusted such that only part of the states contain the true synchronization times and the remaining part is estimated. It should also be taken into account that the rescheduling takes time, during which the system continues. The jobs that occur during this rescheduling, should be introduced as equality constraints in the rescheduling.

- While comparing the SSMPL model to the optimal and nominal SMPL cases, the variable t_r of the Monte Carlo performance index is formulated as:

$$t_r = \max_i x_i(k + N_p) \quad \text{for} \quad i \in (1, ..., N). \tag{7-8}$$

  From analyzing the SMPL system, it is concluded that the best performance index is a trade-off between minimizing the maximum and the sum of the last QCs. It is therefore suggested that t_r should be formulated as:

$$t_r = \sum_i x_i(k + N_p) \quad \text{for} \quad i \in (1, ..., N),$$

  or it should be a combination of the two equations discussed.

- An important direction for future research is the implementation of the Benders' decomposition method. If this method works properly, it has a great advantage over the MILP approach in solving the SSMPL. The (hard to solve) binary variables are decoupled from the noisy handling times. Since the multiple realizations only occur in the dual slave problem, of which the optimization is very fast, more realizations might be implemented than is possible in the Monte Carlo algorithm. A possible explanation to the bad behavior in this work, is suggested by Fischetti et al. (2015) [48]. They argue that the convergence behavior of the optimization heavily depends on the method used for generating the next point to cut. These methods can have good theoretical convergence properties, but fail to be efficient when only one single cut has to be generated. They suppose that the iteration method is weak in polyhedral terms, because the role of the cut is not to let integer points emerge as vertices of the LP relaxation, but to exclude a large subspace from further considerations. They propose multiple solutions, using different cutting schemes that stabilize the cut loops.

- The distribution of the QC handling times and AGV travel times should be determined by obtaining test data, which should improve the accuracy of the model.

- The handling times of the YC can be estimated better. Instead of assigning a constant variable, it should depend on the location at which a container is stored or retrieved. These locations are available beforehand and implementation should improve the model accuracy.

- The rescheduling horizon can be defined in a number of jobs, which is discussed in this work, or in time, which still should be considered.

- The optimizations variables could be parameterized, reducing the number of variables that need to be optimized by the MILP solver [21]. This might result in faster computation times. An example would be parameterizing $\{c_1 \dots c_{40}\}$ for the AGV selection by $2^{w-1} < 40 < 2^w \rightarrow w = 6$ newly defined max-plus binary variables with their adjoint values.

The following recommendations are made on the model implementation and suggestions are done for practicality.

- It might be beneficial to model the system from the cranes' perspective, assigning k to every cycle that a QC or YC handles a container. This approach would introduce more event counters k. On the other hand, both cycle types 1 and 3 ("SYSY" and "YSYS") would have the same dynamics for one event k and are thus equal. Moreover, introducing the possibility to transport multiple containers on one AGV would become less troublesome. Instead of coupling a QC cycle $k + \mu_Q$ to some YC cycle $k + \mu_Y$, four cycles should be coupled, e.g. $k + \mu_{Q,1}$, $k + \mu_{Q,2}$, $k + \mu_{Y,1}$ and $k + \mu_{Y,2}$.
- It might be possible to reduce the number of constraints imposed by the model equations. Four cycle types are introduced, resulting in a fast growing problem size. The time that a quay crane (QC) takes to (un)load an AGV is separated from the time that a YC needs to (un)load an AGV. Also in e.g. cycle type 4 ("YSSY") this time is taken twice, representing "unloading AGV" and "loading AGV" (Eq. (4-14)). These differences are relatively small and might be irrelevant when the noise in the system is considered. Their relevance should be verified and it should be considered if these small differences might be better modeled as noise. If this is feasible, it might decrease the problem size tremendously, which removes the memory issues and possibly decreases solution times.
- When on-line scheduling is implemented correctly, it is possible to evaluate and take advantage of the true data. While the schedule is running, the mean of the distribution should be adjusted to that of the true data, obtaining a better estimate.
- Since the solution time of a schedule fluctuates in some cases, it might be safe to calculate a short prediction horizon, ensuring that the calculation time never exceeds the scheduled time. The calculation of this short prediction horizon can be done in every rescheduling, or can kick in when the initial calculation exceeds a certain time.
- It is recommended to optimize the implementation of the algorithms. A certain ordering of the constraints which are supplied to the MILP, can reduce the time that the solver needs to optimize the schedule. Furthermore, it is important to write the code in such a way that the programming language computes and handles the large matrices fast, as can be seen in Appendix L.

Appendix

Appendix A

Additional Theory on Max-plus Algebra

The properties of the max-plus operators and the switching rule for SMPL are discussed in this appendix.

A-1 Algebraic properties of the max-plus operators

When calculations are done on the system described in the max-plus algebra, it is important to know the properties of the operators $\oplus$ and $\otimes$. The following properties will be of importance in this thesis [26]:

- Associativity:

$$\begin{aligned} \forall x, y, z \in \mathbb{R}_\varepsilon : \quad & x \oplus (y \oplus z) = (x \oplus y) \oplus z, \\ \forall x, y, z \in \mathbb{R}_\varepsilon : \quad & x \otimes (y \otimes z) = (x \otimes y) \otimes z. \end{aligned}$$

- Commutativity:

$$\begin{aligned} \forall x, y \in \mathbb{R}_\varepsilon : \quad & x \oplus y = y \oplus x, \\ \forall x, y \in \mathbb{R}_\varepsilon : \quad & x \otimes y = y \otimes x. \end{aligned}$$

- Distributivity of $\otimes$ over $\oplus$:

$$\forall x, y, z \in \mathbb{R}_\varepsilon : \quad x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z).$$

- Existence of a zero element:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \oplus \varepsilon = \varepsilon \oplus x = x.$$

- Existence of an identity element:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \otimes e = e \otimes x = x.$$

- The zero is absorbing for $\otimes$:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon.$$

- Idempotent of $\oplus$:

$$\forall x \in \mathbb{R}_\varepsilon : \quad x \oplus x = x.$$

A-2 Switching rule

The moments of switching are determined by a switching mechanism. A switching variable $z(k)$ is defined in Eq. (A-1), which can depend on the previous state $\mathbf{x}(k-1)$, previous mode $\ell(k-1)$, the input variable $\mathbf{u}(k)$ and/or an additional control variable $\mathbf{v}(k)$:

$$\mathbf{z}(k) = \Phi(\mathbf{x}(k-1), \ell(k-1), \mathbf{u}(k), \mathbf{v}(k)) \qquad \in \mathbb{R}^{n_z}. \tag{A-1}$$

The space $\mathbb{R}^{n_z}$ can be partitioned in n_m subsets $\mathbb{Z}^{(i)}$, $i = 1, ..., n_m$. For every different mode $\ell(k)$, k is in a different set $\mathbf{z}(k)$. Switching will be necessary in the optimization, but can also be used when a large delay occurs and when changing the job order of cranes can resolve this problem.

Appendix B

Max-plus Binary Control Variables

In this appendix, an example is provided to clarify the use of max-plus binary control variables in the models that are designed.

Example B-6. *Suppose there are two QCs ($n = 2$), three YCs ($m = 3$), three AGVs ($V = 3$), 5 container jobs that need to be handled ($N_p = 5$) and 5 previous containers are considered. The latter results in* $P = \begin{bmatrix} p(k+1) & p(k+2) & p(k+3) & p(k+4) & p(k+5) \end{bmatrix} = \begin{bmatrix} 3 & 4 & 5 & 5 & 5 \end{bmatrix}$*, since there are V initial cycles k. The handling times are as follows:*

- $\tau_2 = 100$*: time QC stores / retrieves container in / from ship,*
- $\tau_3 = 20$*: time QC (un)loads AGV,*
- $\tau_5 = 30$*: time YC (un)loads AGV,*
- $\tau_6 = 270$*: time YC stores / retrieves container in / from yard,*
- $T = \left[\begin{array}{c|ccc} & YC_1 & YC_2 & YC_3 \\ \hline QC_1 & 140 & 100 & 140 \\ QC_2 & 210 & 170 & 130 \end{array}\right]$*: AGV travel time matrix*

Further, the work queue is given in Table G-1 and the initial state:

k	*QC*	*YC*	*cycle type*	*AGV*
-2	*1*	*3*	*D*	*1*
-1	*2*	*1*	*D*	*2*
0	*1*	*2*	*D*	*3*
1	*2*	*3*	*L*	*?*
2	*1*	*2*	*D*	*?*
3	*2*	*1*	*L*	*?*
4	*1*	*2*	*L*	*?*
5	*2*	*1*	*D*	*?*

Table B-1: Work queue of this example

$$x_{init} = \left[\begin{array}{ccc} k-2 & k-1 & k \\ \hline 90 & 90 & 180 \\ 0 & 90 & 90 \\ 0 & 210 & 210 \\ 0 & 0 & 300 \\ 290 & 290 & 290 \end{array}\right].$$

For $k = 1$:

$$
\begin{aligned}
a &= \begin{bmatrix} a_1 & a_2 \end{bmatrix} && = \begin{bmatrix} \varepsilon & 0 \end{bmatrix} && \textit{since } QC_2 \textit{ serves job } (k+1)\\
b &= \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} && = \begin{bmatrix} \varepsilon & \varepsilon & 0 \end{bmatrix} && \textit{since } YC_3 \textit{ serves job } (k+1)\\
m_Q &= \begin{bmatrix} m_{Q_1} & m_{Q_2} & m_{Q_3} \end{bmatrix} && = \begin{bmatrix} \varepsilon & 0 & \varepsilon \end{bmatrix} && \textit{since } QC \textit{ performed in } (k-1)\\
m_Y &= \begin{bmatrix} m_{Y_1} & m_{Y_2} & m_{Y_3} \end{bmatrix} && = \begin{bmatrix} \varepsilon & \varepsilon & 0 \end{bmatrix} && \textit{since } YC \textit{ performed in } (k-2)\\
d_Q &= \begin{bmatrix} d_{Q_1} & d_{Q_2} & d_{Q_3} & d_{Q_4} \end{bmatrix} && = \begin{bmatrix} \varepsilon & 0 & \varepsilon & \varepsilon \end{bmatrix} && \textit{since } QC_2 \textit{ has cycle type } 2\\
d_Y &= \begin{bmatrix} d_{Y_1} & d_{Y_2} & d_{Y_3} & d_{Y_4} \end{bmatrix} && = \begin{bmatrix} \varepsilon & 0 & \varepsilon & \varepsilon \end{bmatrix} && \textit{since } YC_3 \textit{ has cycle type } 2.
\end{aligned}
$$

Now there exists the possibility to select the AGV that terminated previous cycle at either YC_1, YC_2 or YC_3. The work queue shows that $k+1$ is a loading job. Therefore, it is convenient to select the AGV that is already at YC_3, thus AGV_1. It is then obtained that:

$$
\begin{aligned}
c &= \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} && = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 \end{bmatrix} && \textit{since } AGV_1 \textit{ comes from the fifth crane}\\
m_A &= \begin{bmatrix} m_{A_1} & m_{A_2} & m_{A_3} \end{bmatrix} && = \begin{bmatrix} \varepsilon & \varepsilon & 0 \end{bmatrix} && \textit{since } AGV_1 \textit{ served in } (k-2)\\
d_A &= \begin{bmatrix} d_{A_1} & d_{A_2} & d_{A_3} & d_{A_4} \end{bmatrix} && = \begin{bmatrix} \varepsilon & 0 & \varepsilon & \varepsilon \end{bmatrix} && \textit{since } AGV_1 \textit{ has cycle type } 2.
\end{aligned}
$$

It now is possible to calculate the value of $k+1$:

$$
\begin{bmatrix} x_{Q_1}(k+1) \\ x_{Q_2}(k+1) \\ x_{Y_1}(k+1) \\ x_{Y_2}(k+1) \\ x_{Y_3}(k+1) \end{bmatrix} \geq
\begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \tau_4(k+1)+\tau_5(k+1)+ a_2(k+1)+b_3(k+1)+d_{Y_2}(k+1) \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes
\begin{bmatrix} x_{Q_1}(k+1) \\ x_{Q_2}(k+1) \\ x_{Y_1}(k+1) \\ x_{Y_2}(k+1) \\ x_{Y_3}(k+1) \end{bmatrix} \oplus
$$

$$
\begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix} \otimes
\begin{bmatrix} x_{Q_1}(k) \\ x_{Q_2}(k) \\ x_{Y_1}(k) \\ x_{Y_2}(k) \\ x_{Y_3}(k) \end{bmatrix} \oplus
$$

$$
\begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_3(k-\mu_Q)+ a_2(k+1)+m_{Q_2}(k+1)+d_{Q_2}(k+1) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes
\begin{bmatrix} x_{Q_1}(k-1) \\ x_{Q_2}(k-1) \\ x_{Y_1}(k-1) \\ x_{Y_2}(k-1) \\ x_{Y_3}(k-1) \end{bmatrix} \oplus
$$

$$
\begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_1(k+1)+\tau_5(k-\mu_A)+ c_3(k+1)+m_{A_3}(k+1)+d_{A_2}(k+1) & \varepsilon & \tau_5(k-\mu_Y)+\tau_6(k-\mu_Y)+\tau_6(k+1)+ b_3(k+1)+m_{Y_3}(k+1)+d_{Y_2}(k+1) \end{bmatrix} \otimes
\begin{bmatrix} x_{Q_1}(k-2) \\ x_{Q_2}(k-2) \\ x_{Y_1}(k-2) \\ x_{Y_2}(k-2) \\ x_{Y_3}(k-2) \end{bmatrix}
$$

$$\geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 200+15 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{Q_1}(k+1) \\ x_{Q_2}(k+1) \\ x_{Y_1}(k+1) \\ x_{Y_2}(k+1) \\ x_{Y_3}(k+1) \end{bmatrix} \oplus \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix} \otimes \begin{bmatrix} 180 \\ 90 \\ 210 \\ 300 \\ 290 \end{bmatrix} \oplus$$
$$\begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 10 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 90 \\ 90 \\ 210 \\ 0 \\ 290 \end{bmatrix} \oplus \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0+15 & \varepsilon & 15+160+160 \end{bmatrix} \otimes \begin{bmatrix} 90 \\ 0 \\ 0 \\ 0 \\ 290 \end{bmatrix}$$

$$\geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 215 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{Q_1}(k+1) \\ x_{Q_2}(k+1) \\ 210 \\ 300 \\ 625 \end{bmatrix} \oplus \begin{bmatrix} 180 \\ 100 \\ 210 \\ 300 \\ 625 \end{bmatrix}$$

$$\geq \begin{bmatrix} 180 \\ 840 \\ 210 \\ 300 \\ 625 \end{bmatrix}$$

For $k+2$:

$$\begin{aligned} a &= \begin{bmatrix} a_1 & a_2 \end{bmatrix} &&= \begin{bmatrix} 0 & \varepsilon \end{bmatrix} \\ b &= \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} &&= \begin{bmatrix} \varepsilon & 0 & \varepsilon \end{bmatrix} \\ m_Q &= \begin{bmatrix} m_{Q_1} & m_{Q_2} & m_{Q_3} \end{bmatrix} &&= \begin{bmatrix} \varepsilon & 0 & \varepsilon \end{bmatrix} \\ m_Y &= \begin{bmatrix} m_{Y_1} & m_{Y_2} & m_{Y_3} \end{bmatrix} &&= \begin{bmatrix} \varepsilon & 0 & \varepsilon \end{bmatrix} \\ d_Q &= \begin{bmatrix} d_{Q_1} & d_{Q_2} & d_{Q_3} & d_{Q_4} \end{bmatrix} &&= \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \\ d_Y &= \begin{bmatrix} d_{Y_1} & d_{Y_2} & d_{Y_3} & d_{Y_4} \end{bmatrix} &&= \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \end{aligned}$$

Now there exists the possibility to select the AGV that terminated previous cycle at either YC_1, YC_2 or QC_2. Suppose, AGV_2 (coming from YC_1) is selected to serve this job, then:

$$\begin{aligned} c &= \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} &&= \begin{bmatrix} \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \end{bmatrix} \\ m_A &= \begin{bmatrix} m_{A_1} & m_{A_2} & m_{A_3} & m_{A_4} \end{bmatrix} &&= \begin{bmatrix} \varepsilon & \varepsilon & 0 & \varepsilon \end{bmatrix} \\ d_A &= \begin{bmatrix} d_{A_1} & d_{A_2} & d_{A_3} & d_{A_4} \end{bmatrix} &&= \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \end{aligned}$$

It is verifiable that:

$$\begin{bmatrix} x_{Q_1}(k+2) \\ x_{Q_2}(k+2) \\ x_{Y_1}(k+2) \\ x_{Y_2}(k+2) \\ x_{Y_3}(k+2) \end{bmatrix} \geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_3(k+2)+\tau_4(k+2)+ & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ a_1(k+2)+b_2(k+2)+d_{Y_1}(k+2) & & & & \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{Q_1}(k+2) \\ x_{Q_2}(k+2) \\ x_{Y_1}(k+2) \\ x_{Y_2}(k+2) \\ x_{Y_3}(k+2) \end{bmatrix} \oplus \mathrm{E} \otimes \begin{bmatrix} x_{Q_1}(k+1) \\ x_{Q_2}(k+1) \\ x_{Y_1}(k+1) \\ x_{Y_2}(k+1) \\ x_{Y_3}(k+1) \end{bmatrix} \oplus$$

$$\begin{bmatrix} \tau_2(k+2)+\tau_3(k-\mu_Q)+ & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ a_1(k+2)+m_{Q_2}(k+2)+d_{Q_1}(k+2) & & & & \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_5(k-\mu_Y)+\tau_6(k-\mu_Y)+ & \varepsilon \\ & & & b_2(k+2)+m_{Y_2}(k+2)+d_{Y_1}(k+2) & \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{Q_1}(k) \\ x_{Q_2}(k) \\ x_{Y_1}(k) \\ x_{Y_2}(k) \\ x_{Y_3}(k) \end{bmatrix} \oplus$$

$$\begin{bmatrix} \varepsilon & \varepsilon & \tau_1(k+2)+\tau_5(k-\mu_A)+ & \varepsilon & \varepsilon \\ & & c_4(k+2)+m_{A_2}(k+2)+d_{A_1}(k+2) & & \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{Q_1}(k-1) \\ x_{Q_2}(k-1) \\ x_{Y_1}(k-1) \\ x_{Y_2}(k-1) \\ x_{Y_3}(k-1) \end{bmatrix} \oplus \mathcal{E} \otimes \begin{bmatrix} x_{Q_1}(k-2) \\ x_{Q_2}(k-2) \\ x_{Y_1}(k-2) \\ x_{Y_2}(k-2) \\ x_{Y_3}(k-2) \end{bmatrix}$$

$$\geq \begin{bmatrix} (10+80+180) \oplus (15+130+210) \\ 840 \\ 210 \\ (10+110+x_{Q_1}(k+2)) \oplus (15+160+300) \\ 625 \end{bmatrix}$$

$$\geq \begin{bmatrix} 355 \\ 840 \\ 210 \\ 475 \\ 625 \end{bmatrix}.$$

Following the same fashion, $k+3, k+4$ and $k+5$ can be calculated. Suppose AGV_3, AGV_2 and AGV_1 are chosen, respectively. Then the states evolve as:

$$\begin{bmatrix} k+3 & k+4 & k+5 \\ \hline 355 & 935 & 935 \\ 930 & 930 & 1100 \\ 545 & 545 & 1220 \\ 475 & 810 & 810 \\ 625 & 625 & 625 \end{bmatrix} \tag{B-1}$$

dit overlapt met appendix hand calculations: richt dit deel op binaire variabelen of haal k+3 tot k+5 weg. △

Appendix C

Example of MPL models

In this appendix an MPL, SMPL and SSMPL model for a production system are proposed.

Example C-7. *To follow a logical procedure, this example is divided into three parts. First, an MPL system is created. Secondly, a new mode is added and thus an SMPL system is then obtained. Lastly, the uncertainty in the parameters is considered and the SMPL system is extended to an SSMPL system.*

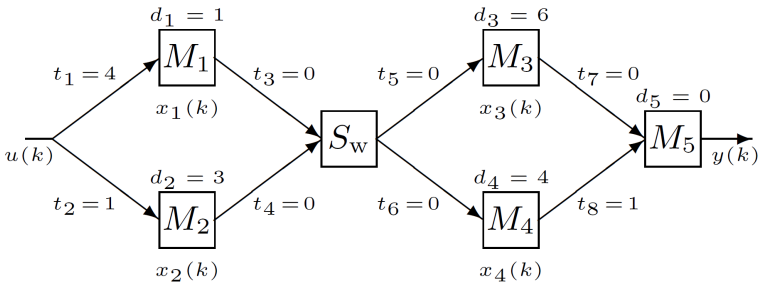

Figure C-1: The production system with five machines that is considered in Example (C-7).

The MPL system

A production system exists of five machines: M_1, M_2, M_3, M_4 *and* M_5*. The input is fed into machines* M_1 *and* M_2*. The products go from* M_1 *to* M_3 *and from* M_2 *to* M_4*. Both products will be assembled instantaneously in* M_5*, see Figure C-1.* $u(k)$ *is the time instant at which the system is fed for the* k^{th} *time,* $y(k) = x_5(k)$ *is the time instant at which the* k^{th} *product leaves the system and* $x_i(k)$ *is the time instant at which machine* i *starts a process for the* k^{th} *time. The variables* t_j *are the transportation times and* d_i *the processing times. It is assumed that each machine starts working as soon as possible on each batch. For example,* M_1 *can start working when*

1. *it has finished its previous batch:*

$$x_1(k) \geq x_1(k-1) + d_1,$$

2. *the new material has been transported to the machine:*

$$x_1(k) \geq u(k) + t_1.$$

One of these two conditions takes more time than the other and should become equality:

$$x_1(k) = \max(x_1(k-1) + d_1, u(k) + t_1).$$

In the same fashion, the system equations of x_2, x_3, x_4 and x_5 could be calculated:

$$x_2(k) = \max(x_2(k-1) + d_2, u(k) + t_2)$$

and

$$\begin{aligned}
x_3(k) &= \max(x_3(k-1) + d_3, x_1(k) + d_1 + t_3 + t_5) \\
&= \max(x_3(k-1) + d_3, x_1(k-1) + 2d_1 + t_3 + t_5, u(k) + d_1 + t_1 + t_3 + t_5) \\
x_4(k) &= \max(x_4(k-1) + d_4, x_2(k) + d_2 + t_4 + t_6) \\
&= \max(x_4(k-1) + d_4, x_2(k-1) + 2d_2 + t_4 + t_6, u(k) + d_2 + t_2 + t_4 + t_6) \\
x_5(k) &= \max(x_3(k) + d_3 + t_7, x_4(k) + d_4 + t_8) \\
&= \max(x_3(k-1) + 2d_3 + t_7, x_1(k-1) + 2d_1 + d_3 + t_3 + t_5 + t_7, \\
&\qquad u(k) + d_1 + d_3 + t_1 + t_3 + t_5 + t_7, \\
&\qquad x_4(k-1) + 2d_4 + t_8, x_2(k-1) + 2d_2 + d_4 + t_4 + t_6 + t_8, \\
&\qquad u(k) + d_2 + d_4 + t_2 + t_4 + t_6 + t_8).
\end{aligned}$$

When these resulting equations are written in max-plus algebra, such as in Eq. (3-5) and Eq. (3-6), the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ can be obtained:

$$\mathbf{A} = \begin{bmatrix}
d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon & \\
2d_1 + t_3 + t_5 & \varepsilon & d_3 & \varepsilon & \varepsilon & \\
\varepsilon & 2d_2 + t_4 + t_6 & \varepsilon & d_4 & \varepsilon & \\
2d_1 + d_3 + t_3 + t_5 + t_7 & 2d_2 + d_4 + t_4 + t_6 + t_8 & 2d_3 + t_7 & 2d_4 + t_8 & \varepsilon &
\end{bmatrix}$$

$$= \begin{bmatrix}
1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\
2 & \varepsilon & 6 & \varepsilon & \varepsilon \\
\varepsilon & 6 & \varepsilon & 4 & \varepsilon \\
8 & 11 & 12 & 9 & \varepsilon
\end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix}
t_1 \\ t_2 \\ d_1 + t_1 + t_3 + t_5 \\ d_2 + t_2 + t_4 + t_6 \\ \max(d_1 + d_3 + t_1 + t_3 + t_5 + t_7, d_2 + d_4 + t_2 + t_4 + t_6 + t_8)
\end{bmatrix} = \begin{bmatrix}
4 \\ 1 \\ 5 \\ 4 \\ 11
\end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 8 & 11 & 12 & 9 & \varepsilon \end{bmatrix}.$$

The MPL system is now obtained.

The SMPL system

Now suppose that the system is interconnected and the batches of machines M_1 and M_2 can proceed to either M_3 or M_4, depending on which batch is completed first. The products coming from machine M_1 and M_2 are directed to a switching device S_w that feeds the first product in the k^{th} cycle to the slowest machine. Since the processing time of M_3 is longer than that of M_4, it is fed to M_3. The second batch is fed to the fastest machine M_4 to reduce the overall production time. Now there exist two scenarios; the first one when M_1 is fed to M_3: $x_1(k) + d_1 \leq x_2(k) + d_2$, the second one when M_1 is fed to M_4: $x_1(k) + d_1 > x_2(k) + d_2$. The first scenario will be referred to as "mode $\ell = 1$", the second scenario will be referred to as "mode $\ell = 2$".

The first mode is already considered in the first part of this example, except now $\mathbf{A} = \mathbf{A}^{(1)}$, $\mathbf{B} = \mathbf{B}^{(1)}$ and $\mathbf{C} = \mathbf{C}^{(1)}$. The second mode results in the following equations:

$$
\begin{aligned}
x_1(k) &= \max(x_1(k-1) + d_1, u(k) + t_1) \\
x_2(k) &= \max(x_2(k-1) + d_2, u(k) + t_2) \\
x_3(k) &= \max(x_3(k-1) + d_3, x_2(k) + d_2 + t_4 + t_5) \\
&= \max(x_3(k-1) + d_3, x_2(k-1) + 2d_2 + t_4 + t_5, u(k) + d_2 + t_2 + t_4 + t_5) \\
x_4(k) &= \max(x_4(k-1) + d_4, x_1(k) + d_1 + t_3 + t_6) \\
&= \max(x_4(k-1) + d_4, x_1(k-1) + 2d_1 + t_3 + t_6, u(k) + d_1 + t_1 + t_3 + t_6) \\
x_5(k) &= \max(x_3(k) + d_3 + t_7, x_4(k) + d_4 + t_8) \\
&= \max(x_3(k-1) + 2d_3 + t_7, x_2(k-1) + 2d_2 + d_3 + t_4 + t_5 + t_7, \\
&\qquad u(k) + d_2 + d_3 + t_2 + t_4 + t_5 + t_7, \\
&\qquad x_4(k-1) + 2d_4 + t_8, x_1(k-1) + 2d_1 + d_4 + t_3 + t_6 + t_8, \\
&\qquad u(k) + d_1 + d_4 + t_1 + t_3 + t_6 + t_8).
\end{aligned}
$$

When these equations are again written in max-plus algebra, the system matrices can be obtained for the second mode of the SMPL system:

$$
\begin{aligned}
\mathbf{A}^{(2)} &= \begin{bmatrix}
d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon & \\
\varepsilon & 2d_2 + t_4 + t_5 & d_3 & \varepsilon & \varepsilon & \\
2d_1 + t_3 + t_6 & \varepsilon & \varepsilon & d_4 & \varepsilon & \\
2d_1 + d_4 + t_3 + t_6 + t_8 & 2d_2 + d_3 + t_4 + t_5 + t_7 & 2d_3 + t_7 & 2d_4 + t_8 & \varepsilon &
\end{bmatrix} \\
&= \begin{bmatrix}
1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & 6 & 6 & \varepsilon & \varepsilon \\
2 & \varepsilon & \varepsilon & 4 & \varepsilon \\
7 & 12 & 12 & 9 & \varepsilon
\end{bmatrix}
\end{aligned}
$$

$$\mathbf{B}^{(2)} = \begin{bmatrix} t_1 \\ t_2 \\ d_2 + t_2 + t_4 + t_5 \\ d_1 + t_1 + t_3 + t_6 \\ \max(d_2 + d_3 + t_2 + t_4 + t_5 + t_7, d_1 + d_4 + t_1 + t_3 + t_6 + t_8) \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 4 \\ 5 \\ 10 \end{bmatrix}$$

$$\mathbf{C}^{(2)} = \begin{bmatrix} 7 & 12 & 12 & 9 & \varepsilon \end{bmatrix}.$$

The SMPL system with $\ell = \{1, 2\}$ is now obtained.

The SSMPL system

In the SMPL model, the system matrices contain d_i as well as on t_j. Suppose the switching between modes is deterministic and the processing time and the transportation time are both stochastic due to variable transportation times t_j and processing times d_i. Furthermore, suppose there are no constant deviations, so no model errors. Now it cannot be assumed that d_i and t_j are constant for every k, but they should depend on k: $d_i(k)$ and $t_j(k)$. The equations for both modes:

$$\begin{aligned} x_1^{(1,2)}(k) &= \max(x_1(k-1) + d_1(k-1), u(k) + t_1(k)) \\ x_2^{(1,2)}(k) &= \max(x_2(k-1) + d_2(k-1), u(k) + t_2(k)). \end{aligned}$$

Furthermore for $\ell = 1$:

$$\begin{aligned} x_3^{(1)}(k) &= \max(x_3(k-1) + d_3(k-1), x_1(k) + d_1(k) + t_3(k) + t_5(k)) \\ &= \max(x_3(k-1) + d_3(k-1), \\ &\qquad x_1(k-1) + d_1(k-1) + d_1(k) + t_3(k) + t_5(k), \\ &\qquad u(k) + d_1(k) + t_1(k) + t_3(k) + t_5(k)) \\ x_4^{(1)}(k) &= \max(x_4(k-1) + d_4(k-1), x_2(k) + d_2(k) + t_4(k) + t_6(k)) \\ &= \max(x_4(k-1) + d_4(k-1), \\ &\qquad x_2(k-1) + d_2(k-1) + d_2(k) + t_4(k) + t_6(k), \\ &\qquad u(k) + d_2(k) + t_2(k) + t_4(k) + t_6(k)) \\ x_5^{(1)}(k) &= \max(x_3(k) + d_3(k) + t_7(k), x_4(k) + d_4(k) + t_8(k)) \\ &= \max(x_3(k-1) + d_3(k-1) + d_3(k) + t_7(k), \\ &\qquad x_1(k-1) + d_1(k-1) + d_1(k) + d_3(k) + t_3(k) + t_5(k) + t_7(k), \\ &\qquad u(k) + d_1(k) + d_3(k) + t_1(k) + t_3(k) + t_5(k) + t_7(k), \\ &\qquad x_4(k-1) + d_4(k-1) + d_4(k) + t_8(k), \\ &\qquad x_2(k-1) + d_2(k-1) + d_2(k) + d_4 + t_4(k) + t_6(k) + t_8(k), \\ &\qquad u(k) + d_2(k) + d_4(k) + t_2(k) + t_4(k) + t_6(k) + t_8(k)) \end{aligned}$$

and for $\ell = 2$:

$$
\begin{aligned}
x_3^{(2)}(k) &= \max(x_3(k-1) + d_3(k-1), x_2(k) + d_2(k) + t_4(k) + t_5(k)) \\
&= \max(x_3(k-1) + d_3(k-1), \\
&\qquad x_2(k-1) + d_2(k-1) + d_2(k) + t_4(k) + t_5(k), \\
&\qquad u(k) + d_2(k) + t_2(k) + t_4(k) + t_5(k)) \\
x_4^{(2)}(k) &= \max(x_4(k-1) + d_4(k-1), x_1(k) + d_1(k) + t_3(k) + t_6(k)) \\
&= \max(x_4(k-1) + d_4(k-1), \\
&\qquad x_1(k-1) + d_1(k-1) + d_1(k) + t_3(k) + t_6(k), \\
&\qquad u(k) + d_2(k) + t_1(k) + t_3(k) + t_6(k)) \\
x_5^{(2)}(k) &= \max(x_3(k) + d_3(k) + t_7(k), x_4(k) + d_4(k) + t_8(k)) \\
&= \max(x_3(k-1) + d_3(k-1) + d_3(k) + t_7(k), \\
&\qquad x_2(k-1) + d_2(k-1) + d_2(k) + d_3(k) + t_4(k) + t_5(k) + t_7(k), \\
&\qquad u(k) + d_2(k) + d_3(k) + t_2(k) + t_4(k) + t_5(k) + t_7(k), \\
&\qquad x_4(k-1) + d_4(k-1) + d_4(k) + t_8(k), \\
&\qquad x_1(k-1) + d_1(k-1) + d_1(k) + d_4(k) + t_3(k) + t_6(k) + t_8(k), \\
&\qquad u(k) + d_1(k) + d_4(k) + t_1(k) + t_3(k) + t_6(k) + t_8(k)).
\end{aligned}
$$

Putting the equations in max-plus-linear notation for both modes $\ell = 1$ and $\ell = 2$ results in the matrices presented below.

System matrices mode $\ell = 1$

The system matrices $\mathbf{A}^{(1)}, \mathbf{B}^{(1)}$ and $\mathbf{C}^{(1)}$ for the first mode $\ell = 1$ for the SSMPL system:

$$
\mathbf{A}^{(1)} = \begin{bmatrix}
d_1(k-1) & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & d_2(k-1) & \varepsilon & \varepsilon & \varepsilon \\
d_1(k-1) + d_1(k) + t_3(k) + t_5(k) & \varepsilon & d_3(k-1) & \varepsilon & \varepsilon \\
\varepsilon & d_2(k-1) + d_2(k) + t_4(k) + t_6(k) & \varepsilon & d_4(k-1) & \varepsilon \\
d_1(k-1) + d_1(k) + d_3(k) + t_3(k) + t_5(k) + t_7(k) & d_2(k-1) + d_2(k) + d_4(k) + t_4(k) + t_6(k) + t_8(k) & d_3(k-1) + d_3(k) + t_7(k) & d_4(k-1) + d_4(k) + t_8(k) & \varepsilon
\end{bmatrix}
$$

$$
\mathbf{B}^{(1)} = \begin{bmatrix}
t_1(k) \\
t_2(k) \\
d_1(k) + t_1(k) + t_3(k) + t_5(k) \\
d_2(k) + t_2(k) + t_4(k) + t_6(k) \\
\max(d_1(k) + d_3(k) + t_1(k) + t_3(k) + t_5(k) + t_7(k), d_2(k) + d_4(k) + t_2(k) + t_4(k) + t_6(k) + t_8(k))
\end{bmatrix}
$$

$$
\mathbf{C}^{(1)} = \begin{bmatrix}
d_1(k-1) + d_1(k) + d_3(k) + t_3(k) + t_5(k) + t_7(k) \\
d_2(k-1) + d_2(k) + d_4(k) + t_4(k) + t_6(k) + t_8(k) \\
d_3(k-1) + d_3(k) + t_7(k) \\
d_4(k-1) + d_4(k) + t_8(k) \\
\varepsilon
\end{bmatrix}^T .
$$

System matrices mode $\ell = 2$

The system matrices $\mathbf{A}^{(2)}, \mathbf{B}^{(2)}$ *and* $\mathbf{C}^{(2)}$ *for the second mode* $\ell = 2$ *for the SSMPL system:*

$$\mathbf{A}^{(2)} = \begin{bmatrix} d_1(k-1) & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2(k-1) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2(k-1)+d_2(k)+t_4(k)+t_5(k) & d_3(k-1) & \varepsilon & \varepsilon \\ d_1(k-1)+d_1(k)+t_3(k)+t_6(k) & \varepsilon & \varepsilon & d_4(k-1) & \varepsilon \\ d_1(k-1)+d_1(k)+d_4(k)+t_3(k)+t_6(k)+t_8(k) & d_2(k-1)+d_2(k)+d_3(k)+t_4(k)+t_5(k)+t_7(k) & d_3(k-1)+d_3(k)+t_7(k) & d_4(k-1)+d_4(k)+t_8(k) & \varepsilon \end{bmatrix}$$

$$\mathbf{B}^{(2)} = \begin{bmatrix} t_1(k) \\ t_2(k) \\ d_2(k)+t_2(k)+t_4(k)+t_5(k) \\ d_1(k)+t_1(k)+t_3(k)+t_6(k) \\ \max(d_2(k)+d_3(k)+t_2(k)+t_4(k)+t_5(k)+t_7(k), d_1(k)+d_4(k)+t_1(k)+t_3(k)+t_6(k)+t_8(k)) \end{bmatrix}$$

$$\mathbf{C}^{(2)} = \begin{bmatrix} d_1(k-1)+d_1(k)+d_4(k)+t_3(k)+t_6(k)+t_8(k) \\ d_2(k-1)+d_2(k)+d_3(k)+t_4(k)+t_5(k)+t_7(k) \\ d_3(k-1)+d_3(k)+t_7(k) \\ d_4(k-1)+d_4(k)+t_8(k) \\ \varepsilon \end{bmatrix}^T .$$

All the stochastic, variable parameters can be gathered in the uncertainty vector $\mathrm{E}_u(k)$*:*

$$\mathrm{E}_u(k) = \Big[d_1(k-1) \;\; d_1(k) \;\; d_2(k-1) \;\; d_2(k) \;\; d_3(k-1) \;\; d_3(k) \;\; d_4(k-1) \;\; d_4(k) \;\; \cdots \;\; t_1(k) \;\; t_2(k) \;\; t_3(k) \;\; t_4(k) \;\; t_5(k) \;\; t_6(k) \;\; t_7(k) \;\; t_8(k) \Big]^T .$$

Note that since the variables t_j *and* d_i *are depending on event counter* k*, it is not possible to substitute the values given in Figure C-1 in the system matrices of the SSMPL model.* △

Appendix D

Implicit vs. Explicit Models

In Section 4-1-2 it is stated that an implicit model would result in less constraints in the MILP formulation. In this appendix this statement will be clarified.

Suppose Eq. (4-2)-(4-3), Eq. (4-9)-(4-10), Eq. (4-12)-(4-13) and Eq. (4-14)-(4-15) are all considered for the MILP inequality constraints. Furthermore, only the binary variables $\{d_1, ..., d_4\}$ are taken into account, with which a cycle type for the QC, YC and AGV can be selected. The cycle types are: "SYSY","SYYS","YSYS" and "YSSY". The inequality expressions that are then obtained for the QC synchronization:

$$
\begin{aligned}
&x_Q(k) \geq \tau_2(k) \otimes \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_1} \\
&x_Q(k) \geq \tau_1(k) \otimes \tau_5(k-\mu_A) \otimes x_Y(k-\mu_A) \otimes d_{A_1} \\
&x_Q(k) \geq \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_2} \\
&x_Q(k) \geq \tau_4(k) \otimes \tau_5(k) \otimes x_Y(k) \otimes d_{A_2} \\
&x_Q(k) \geq \tau_2(k-\mu_Q) \otimes \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_3} \\
&x_Q(k) \geq \tau_4(k) \otimes \tau_5(k) \otimes x_Y(k) \otimes d_{A_3} \\
&x_Q(k) \geq \tau_2(k) \otimes \tau_2(k-\mu_Q) \otimes \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_4} \\
&x_Q(k) \geq \tau_1(k) \otimes \tau_3(k-\mu_A) \otimes x_Q(k-\mu_A) \otimes d_{A_4}
\end{aligned}
$$

for the YC synchronization

$$
\begin{aligned}
&x_Y(k) \geq \tau_3(k) \otimes \tau_4(k) \otimes x_Q(k) \otimes d_{A_1} \\
&x_Y(k) \geq \tau_5(k-\mu_Y) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_1} \\
&x_Y(k) \geq \tau_1(k) \otimes \tau_5(k-\mu_A) \otimes x_Y(k-\mu_A) \otimes d_{A_2} \\
&x_Y(k) \geq \tau_5(k-\mu_Y) \otimes \tau_6(k) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_2} \\
&x_Y(k) \geq \tau_1(k) \otimes \tau_3(k-\mu_A) \otimes x_Q(k-\mu_A) \otimes d_{A_3} \\
&x_Y(k) \geq \tau_6(k) \otimes \tau_5(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_3} \\
&x_Y(k) \geq \tau_3(k) \otimes \tau_4(k) \otimes x_Q(k) \otimes d_{A_4} \\
&x_Y(k) \geq \tau_5(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_4}
\end{aligned}
$$

and finally it should hold that

$$x(k) \geq x(k-1).$$

These **seventeen** constraints follow from the implicit model, that is also used for the SSMPL system. The implicit model can be reduced to **fifteen** constraints when assumed that (I) τ_3 and τ_5 are constant variables and (II) if there is no distinction made between the traveling times to and from a departure crane ($\tau_1(k) = \tau_5(k)$). The inequality constraints imposed by the **A**-matrix parameterization for the implicit model are now as follows:

$$x_Q(k) \geq \tau_3 \otimes \tau_2(k) \otimes x_Q(k-\mu_Q) \otimes d_{Q_1} \tag{D-1}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_1(k) \otimes x_Y(k-\mu_A) \otimes d_{A_1} \tag{D-2}$$
$$x_Q(k) \geq \tau_3 \otimes x_Q(k-\mu_Q) \otimes d_{Q_2} \tag{D-3}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_1(k) \otimes x_Y(k) \otimes (d_{A_2} \otimes d_{A_3}) \tag{D-4}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_2(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_3} \tag{D-5}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_2(k) \otimes \tau_2(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_4} \tag{D-6}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_1(k) \otimes x_Q(k-\mu_A) \otimes d_{A_4} \tag{D-7}$$

and

$$x_Y(k) \geq \tau_3 \otimes \tau_1(k) \otimes x_Q(k) \otimes (d_{A_1} \otimes d_{A_4}) \tag{D-8}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_1} \tag{D-9}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_1(k) \otimes x_Y(k-\mu_A) \otimes d_{A_2} \tag{D-10}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_2} \tag{D-11}$$
$$x_Y(k) \geq \tau_3 \otimes \tau_1(k) \otimes x_Q(k-\mu_A) \otimes d_{A_3} \tag{D-12}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k) \otimes x_Y(k-\mu_Y) \otimes d_{Y_3} \tag{D-13}$$
$$x_Y(k) \geq \tau_5 \otimes x_Y(k-\mu_Y) \otimes d_{Y_4}. \tag{D-14}$$

and

$$x(k) \geq x(k-1). \tag{D-15}$$

To obtain the explicit model, the inequalities containing dependence on the current cycle k of the unreduced implicit model are substituted by the following expressions:

- For the SYSY-cycle (d_1):

$$\begin{aligned}\tau_3(k) \otimes \tau_4(k) \otimes x_Q(k) \otimes d_{A_1} \geq \\ \tau_3(k) \otimes \tau_4(k) \otimes \tau_2(k) \otimes \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_1} \oplus \\ \tau_3(k) \otimes \tau_4(k) \otimes \tau_1(k) \otimes \tau_5(k-\mu_A) \otimes x_Y(k-\mu_A) \otimes d_{A_1}.\end{aligned}$$

- For the SYYS-cycle (d_2):

$$\begin{aligned}\tau_4(k) \otimes \tau_5(k) \otimes x_Y(k) \otimes d_{A_2} \geq \\ \tau_4(k) \otimes \tau_5(k) \otimes \tau_1(k) \otimes \tau_5(k-\mu_A) \otimes x_Y(k-\mu_A) \otimes d_{A_2} \oplus \\ \tau_4(k) \otimes \tau_5(k) \otimes \tau_5(k-\mu_Y) \otimes \tau_6(k) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_2}.\end{aligned}$$

- For the YSYS-cycle (d_3):

$$\begin{aligned}\tau_4(k) \otimes \tau_5(k) \otimes x_Y(k) \otimes d_{A_3} \geq \\ \tau_4(k) \otimes \tau_5(k) \otimes \tau_1(k) \otimes \tau_3(k-\mu_A) \otimes x_Q(k-\mu_A) \otimes d_{A_3} \oplus \\ \tau_4(k) \otimes \tau_5(k) \otimes \tau_6(k) \otimes \tau_5(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_3}.\end{aligned}$$

- For the YSSY-cycle (d_4):

$$\begin{aligned}\tau_3(k) \otimes \tau_4(k) \otimes x_Q(k) \otimes d_{A_4} \geq \\ \tau_3(k) \otimes \tau_4(k) \otimes \tau_2(k) \otimes \tau_2(k-\mu_Q) \otimes \tau_3(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_4} \oplus \\ \tau_3(k) \otimes \tau_4(k) \otimes \tau_1(k) \otimes \tau_3(k-\mu_A) \otimes x_Q(k-\mu_A) \otimes d_{A_4}.\end{aligned}$$

The first assumption ($\tau_3(k) = \tau_3, \tau_5(k) = \tau_5$) can also be valid for this model. However, the second assumption can not be used, since τ_1 and τ_4 can exists in one constraint. The inequality constraints imposed by the **A**-matrix parameterization for the explicit model now become:

$$x_Q(k) \geq \tau_3 \otimes \tau_2(k) \otimes x_Q(k-\mu_Q) \otimes d_{Q_1} \tag{D-16}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_1(k) \otimes x_Y(k-\mu_A) \otimes d_{A_1} \tag{D-17}$$
$$x_Q(k) \geq \tau_3 \otimes x_Q(k-\mu_Q) \otimes d_{Q_2} \tag{D-18}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_5 \otimes \tau_1(k) \otimes \tau_4(k) \otimes x_Y(k-\mu_A) \otimes d_{A_2} \tag{D-19}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_5 \otimes \tau_1(k) \otimes \tau_6(k) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_2} \tag{D-20}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_2(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_3} \tag{D-21}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_5 \otimes \tau_1(k) \otimes \tau_4(k) \otimes x_Q(k-\mu_A) \otimes d_{A_3} \tag{D-22}$$
$$x_Q(k) \geq \tau_5 \otimes \tau_5 \otimes \tau_1(k) \otimes \tau_6(k) \otimes x_Y(k-\mu_Y) \otimes d_{Y_3} \tag{D-23}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_2(k) \otimes \tau_2(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_4} \tag{D-24}$$
$$x_Q(k) \geq \tau_3 \otimes \tau_1(k) \otimes x_Q(k-\mu_A) \otimes d_{A_4} \tag{D-25}$$

and

$$x_Y(k) \geq \tau_3 \otimes \tau_3 \otimes \tau_1(k) \otimes \tau_2(k) \otimes x_Q(k-\mu_Q) \otimes d_{Q_1} \tag{D-26}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_3 \otimes \tau_1(k) \otimes \tau_4(k) \otimes x_Y(k-\mu_A) \otimes d_{A_1} \tag{D-27}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_1} \tag{D-28}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_1(k) \otimes x_Y(k-\mu_A) \otimes d_{A_2} \tag{D-29}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k) \otimes \tau_6(k-\mu_Y) \otimes x_Y(k-\mu_Y) \otimes d_{Y_2} \tag{D-30}$$
$$x_Y(k) \geq \tau_3 \otimes \tau_1(k) \otimes x_Q(k-\mu_A) \otimes d_{A_3} \tag{D-31}$$
$$x_Y(k) \geq \tau_5 \otimes \tau_6(k) \otimes x_Y(k-\mu_Y) \otimes d_{Y_3} \tag{D-32}$$
$$x_Y(k) \geq \tau_3 \otimes \tau_3 \otimes \tau_1(k) \otimes \tau_2(k) \otimes \tau_2(k-\mu_Q) \otimes x_Q(k-\mu_Q) \otimes d_{Q_4} \tag{D-33}$$
$$x_Y(k) \geq \tau_3 \otimes \tau_3 \otimes \tau_1(k) \otimes \tau_4(k) \otimes x_Q(k-\mu_A) \otimes d_{A_4} \tag{D-34}$$
$$x_Y(k) \geq \tau_5 \otimes x_Y(k-\mu_Y) \otimes d_{Y_4}. \tag{D-35}$$

and

$$x(k) \geq x(k-1). \tag{D-36}$$

Since there are **21** inequalities describing the MILP constraints for the explicit model, it is already clear that the implicit model (fourteen inequalities) results in less constraints for the MILP problem.

Assuming that there exist P previous cycles, the number of constraints of the implicit model are:

$$nr_{c,impl} = \Big(4NP + 4MP + 4NMP + PN^2 + PM^2 + 2NM + N + M\Big) \times N_{max} \tag{D-37}$$

Assuming that there exist p previous cycles, the number of constraints of the explicit model are:

$$nr_{c,expl} = \Big(4NP + 4MP + 8NMP + 2PN^2 + 2PM^2 + N + M\Big) \times N_{max} \tag{D-38}$$

There is however one downside when the implicit model is used. The number of variables that need to be optimized is namely slightly larger. This is due to the merging of the constraints. In the inequality constraints D-5 and D-9 the variables $(d_{A_2} \otimes d_{A_3})$ and $(d_{A_1} \otimes d_{A_4})$ exist, respectively. In the MILP formulation, these are denoted by the new variables d_{A_5} and d_{A_6}. This on their turn introduce $2 \times N_{max}$ new inequality constraints. However, due to these inequalities, the extra variables directly follow from d_1 to d_4 (Equations 5-8 and 5-9). The number of variables that need to be optimized in respectively the implicit and explicit model:

$$nr_{v,impl} = (N + M + P + 6) \times N_{max} \tag{D-39}$$

$$nr_{v,expl} = (N + M + P + 4) \times N_{max} \tag{D-40}$$

In Table D-1, the number of constraints and variables for all test cases (t.c.) are represented. It can be observed that the difference in the number of constraint far exceeds the difference in the number of optimization variables.

	Nr of constr.		Nr of var.	
t.c.	Impl.	Expl.	Impl.	Expl.
1	4.560	7.125	240	210
2	11.760	19.290	330	300
3	28.950	49.620	435	405
4	38.400	66.120	510	480
5	71.400	126.150	615	585
6	83.910	149.415	630	600
7	136.875	247.695	735	705
8	159.450	288.945	810	780
9	237.675	435.825	915	885

	Nr of constr.		Nr of var.	
t.c.	Impl.	Expl.	Impl.	Expl.
10	299.070	551.040	1005	975
11	415.110	771.270	1110	1080
12	415.110	771.270	1110	1080
13	557.610	1.043.100	1215	1185
14	663.165	1.244.115	1305	1275
15	855.330	1.612.995	1410	1380
16	920.805	1.737.045	1485	1455
17	1.158.030	2.194.125	1590	1560
18	1.816.980	3.468.450	1815	1785

Table D-1: For all eighteen test cases (t.c.), the number of constraints and variables are decided for the implicit as well as the explicit model, assuming that there are P previous cycles.

The same difference becomes clear in Figures D-1 and D-2. The absolute difference in the number of constraints increases, while the absolute difference in variables stays equal (since this is $2 \times N_{max}$). This means that the relative difference in variables reduces.

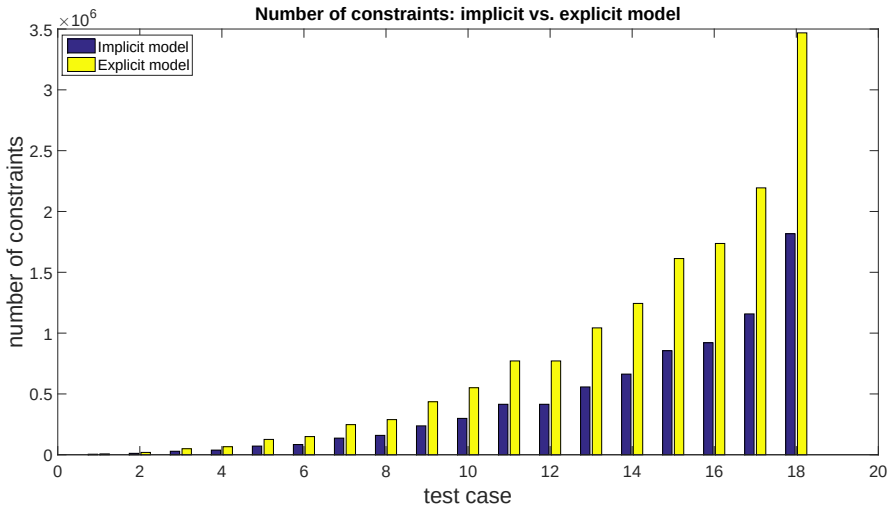

Figure D-1: The number of constraints in the MILP formulation for eighteen test cases for the implicit and explicit model. The problem is always bigger with an explicit model and this problem size grows rapidly.

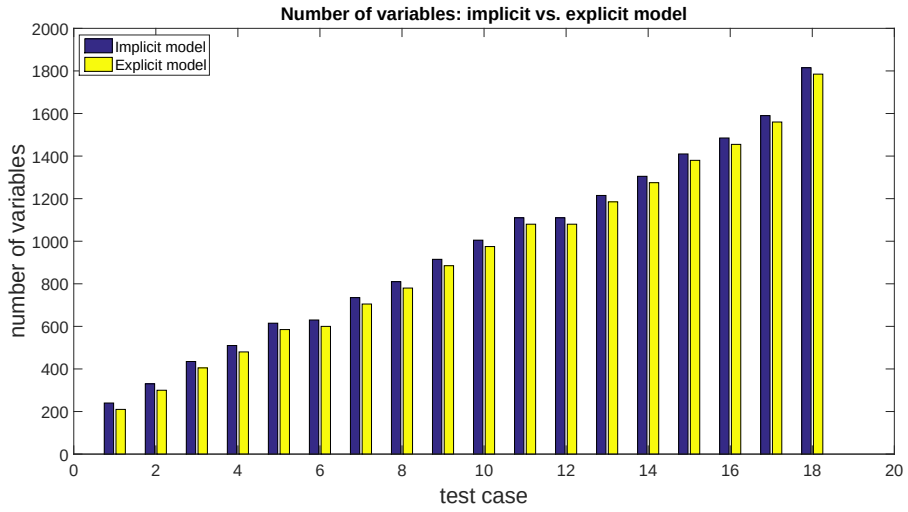

Figure D-2: The number of optimization variables in the MILP formulation for eighteen test cases for the implicit and explicit model. The absolute difference stays the same in all test cases.

The relative difference in constraints reduces, too. This can be seen in Figure D-3. To understand why the relative difference decreases in the beginning and less in the larger test cases, the equations Eq. (D-37) and (D-38) are considered. When the test cases become larger, the following terms have the most weight in the equations:

$$4NMP\ (\approx 50\%), \quad N^2P, \quad M^2P \quad \text{and} \quad 8NMP, \quad 2N^2P, \quad 2M^2P,$$

since have either one squared crane- or two cranes multiplied by P (which grows rapidly). For larger cases, also N^2P and $2N^2P$ have little weight in the final amount of constraints.

Very roughly, the relative difference in the number of constraints is mainly determined by

$$\frac{4NMP + M^2P}{8NMP + 2M^2P} = \frac{4NMP + M^2P}{2 \times (4NMP + M^2P)} = \frac{1}{2}$$

Assuming the relations between the parameters that are defined in the test cases, the relative difference will thus be approximately 0.5.

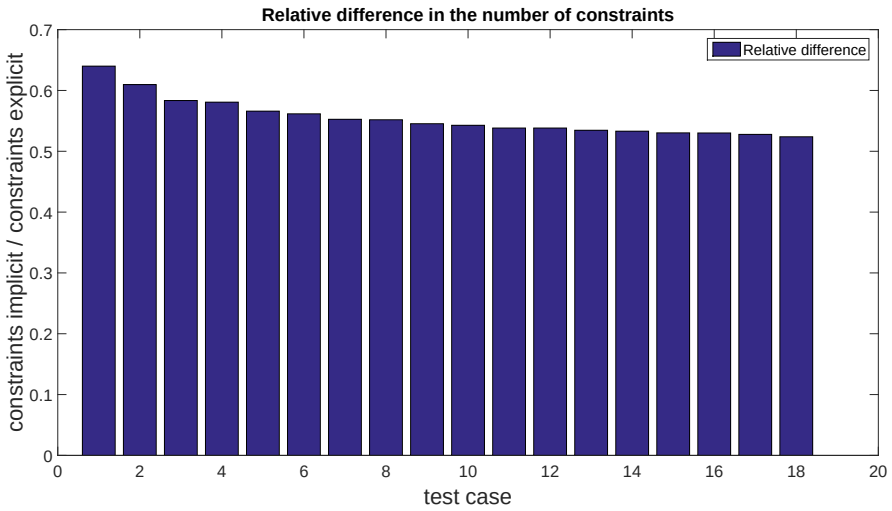

Figure D-3: The relative difference in the number of constraints for the implicit and explicit model. The difference shrinks, but appears to reach a value around 0,5 for the larger test cases.

Appendix E

Sparse Matrices

When an MPL system is formulated in a MILP framework, the amount of zeros in the MILP matrices can be huge. Since the matrices are very large, they take a considerable amount of memory capacity. To reduce this amount as much as possible, the matrices that consist of mostly zeros, should be implemented as sparse matrices. This will reduce the memory usage considerably, as can be seen in Table E-1.

type	**size**	**bytes**	**bytes per element**
diagonal matrix	10.000×10.000	8×10^8	8
diagonal sparse matrix	10.000×10.000	4×10^5	24

Table E-1: Difference in memory usage of a conventional matrix and sparse matrix. The last column shows the bytes used per matrix element.

The differences for the specific case of the container terminal MILP formulation can be seen in Table E-2.

test case	$\mathbf{E}$	$\mathbf{E}_{sparse}$	$\mathbf{F}$	$\mathbf{F}_{sparse}$
1	24.480	5.032	24.480	6.104
8	10.833.680	153.784	2.424.320	170.552
17	303.149.200	1.063.608	32.687.392	1.131.800

Table E-2: Difference in memory usage (bytes) of conventional and sparse matrices for the specific container terminal MILP matrices $\mathbf{E}$ and $\mathbf{F}$.

Appendix F

Search algorithms

MILP solvers use in general the same methodology, of which presolving, heuristics, the branch-and-bound search methods and cutting planes is explained.

Branch-and-bound

At first, the problem is solved by removing all the integrality constraints (those on the binary variables). These constraints allow MIP models to capture the discrete nature of the system. The LP problem that arises, is called the LP relaxation of the original MIP problem. Then a constraint is added of a variable that is restricted to be integer, but whose value in the LP relaxation is fractional. This variable becomes the branching variable and two sub-nodes arise in the search tree (Figure F-1). These sub-nodes both become a new MIP problem, that is solved again by relaxation and the branching of a integer variable. Note that the LP relaxation returns an infeasible solution to the original MILP at first. When a feasible solution is found to the original MIP problem, the node is designated as fathomed (permanent leaf in the search tree). Thereby, the solution is denoted as incumbent. Only if the solution has a better objective, the incumbent is updated by the current incumbent and its objective value. The final optimal solution is found when the difference between the minimum objective value of all current leaves and the incumbent objective value equals zero.

Cutting planes

Cutting planes tighten the formulation by disregarding undesirable fractional solutions, without actually computing them. They do this during the solution process, without creating additional sub-problems. For a more thorough explanation it is referred to the Gurobi documentation [37].

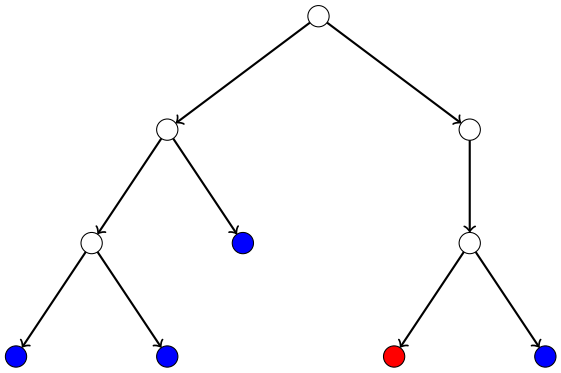

Figure F-1: The search tree where every node is a new MIP which can be branched again. The the leaf nodes denoted in blue are (not yet) branched. The red node is fathomed and a feasible solution for the problem.

Presolve

Presolving reduces the size of the problem by solving variables that follow directly from the constraints and removing the inequality constraints.

Heuristics

Heuristics are used to find good incumbents in a fast manner. This is desirable since (I) it might be impossible to solve the problem to provable optimality if the MIP is too difficult and (II) it helps with the search process and a node being fathomed. If the objective value of the incumbent is lower, the more likely it is that the value of an LP relaxation will exceed it.

Appendix G

Model Verification by Hand Calculations

To verify the model, numerous simulations are verified by hand calculations. In this appendix, an example is provided of such calculation.

Example G-8. *Suppose there are 2 QCs, 3 YCs, 3 AGVs and* $N_p = 5$*. Furthermore* $\tau = \begin{bmatrix} \tau_2 & \tau_3 & \tau_5 & \tau_6 \end{bmatrix} = \begin{bmatrix} 80 & 10 & 15 & 160 \end{bmatrix}$ *and*

$$T = \left[\begin{array}{c|ccc} & YC_1 & YC_2 & YC_3 \\ \hline QC_1 & 130 & 110 & 180 \\ QC_2 & 200 & 140 & 190 \end{array}\right]$$

The work queue and inital states are as follows:

k	***QC***	***YC***	***cycle type***	***AGV***
-3	*1*	*1*	*D*	*1*
-2	*2*	*2*	*D*	*2*
-1	*1*	*3*	*D*	*3*
0	*2*	*1*	*D*	*?*
1	*1*	*2*	*D*	*?*
2	*2*	*3*	*D*	*?*
3	*1*	*1*	*D*	*?*
4	*2*	*2*	*D*	*?*

Table G-1: Work queue and initial AGV dispatching for this example.

$$x_{init} = \left[\begin{array}{ccc} k-1 & k-2 & k-3 \\ \hline 130 & 35 & 35 \\ 35 & 35 & 0 \\ 175 & 175 & 175 \\ 185 & 185 & 10 \\ 320 & 0 & 0 \end{array}\right].$$

For the first cycle k:

$$\begin{aligned}\hat{x}_2(k) \geq\ & \tau_2 + \tau_3 + \hat{x}_2(k-2) \oplus \\ & \tau_v(k) + \tau_5 + \{\hat{x}_3(k-3), \hat{x}_4(k-2), \hat{x}_5(k-1)\} \\ \hat{x}_2(k) \geq\ & 80 + 10 + 35 \oplus \\ & \{200, 140, 190\} + 15 + \{175, 185, 320\}\end{aligned}$$

The values between brackets are respectively $\{AGV_1, AGV_2, AGV_3\}$. AGV_3 would result in a very slow time for the QC synchronization. Furthermore, even though AGV_1 can start traveling soonest, the travel from YC_1 to QC_2 is quite long. Therefore, it would be 50 seconds faster to use AGV_2. Therefore, this latter option is most probably fastest, thus

$$\begin{aligned}\hat{x}_2(k) \geq\ & 125 \ \oplus 140 + 15 + 185 \\ \hat{x}_2(k) \geq\ & 340\end{aligned}$$

and

$$\begin{aligned}\hat{x}_3(k) \geq\ & \tau_3 + \tau_v(k) + \hat{x}_2(k) \oplus \\ & \tau_5 + \tau_6 + \hat{x}_3(k-3) \\ \hat{x}_3(k) \geq\ & 10 + 200 + 340 \oplus 15 + 160 + 175 \\ \hat{x}_3(k) \geq\ & 550\end{aligned}$$

The next job $k+1$ is performed by QC_1 ($\hat{x}_1(k+1)$) and YC_2 ($\hat{x}_4(k+1)$):

$$\begin{aligned}\hat{x}_1(k+1) \geq\ & \tau_2 + \tau_3 + \hat{x}_1(k-1) \oplus \\ & \tau_v(k+1) + \tau_5 + \{\hat{x}_3(k-3), \hat{x}_3(k), \hat{x}_5(k-1)\} \\ \hat{x}_1(k+1) \geq\ & 80 + 10 + 130 \ \oplus \{130, 130, 180\} + 15 + \{175, 550, 320\}\end{aligned}$$

Selecting AGV_1 would result in the fastest synchronization time, obtaining

$$\begin{aligned}\hat{x}_1(k+1) \geq\ & 220 \ \oplus 130 + 15 + 175 \\ \hat{x}_1(k+1) \geq\ & 320\end{aligned}$$

The YC_2 synchronization time then becomes

$$\begin{aligned}\hat{x}_4(k+1) \geq\ & \tau_3 + \tau_v(k+1) + \hat{x}_1(k+1) \oplus \\ & \tau_5 + \tau_6 + \hat{x}_4(k-2) \\ \hat{x}_4(k+1) \geq\ & 10 + 110 + 320 \oplus 15 + 160 + 185 \\ \hat{x}_4(k+1) \geq\ & 440.\end{aligned}$$

For job $k+2$

$$\begin{aligned}\hat{x}_2(k+2) \geq\ & \tau_2 + \tau_3 + \hat{x}_2(k) \oplus \\ & \tau_v(k+2) + \tau_5 + \{\hat{x}_4(k+1), \hat{x}_3(k), \hat{x}_5(k-1)\} \\ \hat{x}_2(k+2) \geq\ & 80 + 10 + 340 \ \oplus \{140, 200, 190\} + 15 + \{440, 550, 320\}.\end{aligned}$$

AGV_2 will not be the best choice. However, between AGV_1 and AGV_3 there is a difference of 70 seconds. Therefore, AGV_3 is chosen.

$$\begin{aligned}\hat{x}_2(k+2) &\geq 430 \oplus 190+15+320\\ \hat{x}_2(k+2) &\geq 525\end{aligned}$$

The YC_3 synchronization time then becomes

$$\begin{aligned}\hat{x}_5(k+2) &\geq \tau_3+\tau_v(k+2)+\hat{x}_2(k+2)\oplus\\ &\quad\ \tau_5+\tau_6+\hat{x}_5(k-1)\\ \hat{x}_5(k+2) &\geq 10+190+525 \oplus 15+160+320\\ \hat{x}_5(k+2) &\geq 725.\end{aligned}$$

For job $k+3$

$$\begin{aligned}\hat{x}_1(k+3) &\geq \tau_2+\tau_3+\hat{x}_1(k+1)\ \oplus\\ &\quad\ \tau_v(k+3)+\tau_5+\{\hat{x}_4(k+1),\hat{x}_3(k),\hat{x}_5(k+2)\}\\ \hat{x}_1(k+3) &\geq 80+10+320\ \oplus\{110,130,180\}+15+\{440,550,725\}.\end{aligned}$$

Choosing AGV_1 gives:

$$\begin{aligned}\hat{x}_1(k+3) &\geq 410 \oplus 110+15+440\\ \hat{x}_1(k+3) &\geq 565\end{aligned}$$

The YC_1 synchronization time then becomes

$$\begin{aligned}\hat{x}_3(k+3) &\geq \tau_3+\tau_v(k+3)+\hat{x}_1(k+3)\oplus\\ &\quad\ \tau_5+\tau_6+\hat{x}_3(k)\\ \hat{x}_3(k+3) &\geq 10+130+565\ \oplus 15+160+550\\ \hat{x}_3(k+3) &\geq 725.\end{aligned}$$

For job $k+4$

$$\begin{aligned}\hat{x}_2(k+4) &\geq \tau_2+\tau_3+\hat{x}_2(k+2)\ \oplus\\ &\quad\ \tau_v(k+4)+\tau_5+\{\hat{x}_4(k+3),\hat{x}_3(k),\hat{x}_5(k+2)\}\\ \hat{x}_2(k+4) &\geq 80+10+525\ \oplus\{200,200,190\}+15+\{725,550,725\}.\end{aligned}$$

Choosing AGV_2 gives:

$$\begin{aligned}\hat{x}_2(k+4) &\geq 615 \oplus 200+15+550\\ \hat{x}_2(k+4) &\geq 765\end{aligned}$$

The YC_2 synchronization time then becomes

$$\begin{aligned}\hat{x}_4(k+4) &\geq \tau_3+\tau_v(k+4)+\hat{x}_2(k+4)\oplus\\ &\quad\ \tau_5+\tau_6+\hat{x}_4(k+1)\\ \hat{x}_4(k+4) &\geq 10+140+765 \oplus 15+160+440\\ \hat{x}_4(k+4) &\geq 915.\end{aligned}$$

The states thus become:

$$x_{init} = \begin{bmatrix} k & k+1 & k+2 & k+3 & k+4 \\ 180 & 320 & 320 & 565 & 565 \\ 340 & 340 & 525 & 525 & 765 \\ 550 & 550 & 550 & 725 & 725 \\ 185 & 440 & 440 & 440 & 915 \\ 320 & 320 & 725 & 725 & 725 \end{bmatrix}$$

and is compared to the result of MATLAB. It is not possible to guarantee the implementation is correct by checking various small test cases. However, since they return the same result as the hand calculations, it can be stated that the implementation is correct with highly probability. This scheme is also visible in Figure 5-4.

△

Appendix H

Job Order Switching Schemes

Switching of job orders most probably decreases the overall makespan. The two proposed switching schemes, full- and threshold switching, are elaborated on next.

Full switching scheme

Example H-9. *There are three jobs: k, k+1 and k+2. In the current formulation:*

$$x(k+2) \geq x(k+1) \qquad\qquad x(k+1) \geq x(k). \tag{H-1}$$

To switch the order of events, the following should all be defined

$$\begin{array}{lll} x(k+1) \geq x(k) \ \rightarrow g_{10} & x(k+2) \geq x(k+1) \ \rightarrow g_{21} & x(k+2) \geq x(k) \ \rightarrow g_{20} \\ x(k) \geq x(k+1) \ \rightarrow \bar{g}_{10} & x(k+1) \geq x(k+2) \ \rightarrow \bar{g}_{21} & x(k) \geq x(k+2) \ \rightarrow \bar{g}_{20} \end{array} \tag{H-2}$$

for every crane. Besides the numerous extra constraints, it introduces three new max-plus binary control variables g_{10}, g_{21} and g_{20} with their complements. △

A more general notation for the inequalities in Eq. (H-2) would be

$$\begin{aligned} x(k+j) &\geq x(k+i) + \tau + w + z + g \qquad j > i \\ x(k+i) &\geq x(k+j) + \tau + w + z + \bar{g} \qquad i > j, \end{aligned}$$

where τ some handling time, w a combination of known - and z a combination of unknown max-plus binary variables.

Example (H-9) shows that the number of constraints and variables will increase when switching is added to the system. Instead of constraints for just the two state-inequalities Eq. (H-1), described in parameterization M (Section 5-1), it needs constraints for all six state-inequalities of Eq. (H-2). The number of these state-inequalities equals $N_{switch} \times (N_{switch} - 1)$, where

N_{switch} the horizon in which job switching is allowed. Note that this concerns the *state-inequalities* and should be multiplied according to Eq. (5-4) to obtain the real number of constraints.

Another crucial point illustrated by Example (H-9) is the increase in control variables. The number of variables necessary equals $\frac{1}{2}\left(N_{switch} \times (N_{switch} - 1)\right)$. These should be added (not multiplied) to the existing set of max-plus binary variables, each selecting a different set of constraints corresponding to different state-inequalities.

Due to the explained growth in constraints, this method will not be advised to implement. Memory issues are already a concern in the suggested SMPL model and it is therefore supposed that the full switching scheme will fail for the large (and possibly the medium) test cases.

Threshold switching

For the threshold switching it is not necessary to introduce more constraints. Instead, the schedule is calculated just as explained before, where after the state is analyzed. When it appears that a crane has a large waiting time, a switch in job order is considered for only a small part of the system. After the switch, the schedule is recalculated to check if the switch results in a faster schedule. The threshold switching looks more promising since it avoids memory issues, but is also convenient since large delays and thus possible switches are uncommon.

Before specifying how the switch would occur, the delays are studied, starting with the QC delays (type II) for which minimum waiting times are crucial. A delay might arise when a QC switches from serving discharge jobs to serving loading jobs. Accordingly, a delay happens only every now and then, since the QC serves the job types in batches (containers are unloaded off the ship, before placing containers back on the vessel). It finishes loading an AGV and starts waiting for its next job, say $(k+j)$. This job starts at the yard side, were the YC starts to retrieve $(k+j)$ after it finished its last job $(k+i^*)$. Hereafter, the container is placed on the AGV (τ_5 seconds) and the AGV transports this container to the quay (τ_v seconds). If there is a delay, the YC should perform job $(k+j)$ earlier and it should be moved up in its job sequence (while leaving the QC job sequence untouched). The ideal time for job $(k+j)$ to leave the yard side is:

$$\hat{x}_{YC,ideal}(k+j) = \hat{x}_{QC,ideal}(k+j) - t_v(k+j) - \tau_5.$$

When there is a delay, $\hat{x}_{YC}(k+j) > \hat{x}_{YC,ideal}(k+j)$ is true and job $(k+j)$ should be substituted directly before the job for which it holds:

$$\hat{x}_{YC}(k+i) \leq \hat{x}_{YC,ideal}(k+j). \tag{H-3}$$

Suppose the job sequence of a YC changes as follows when $(k+j)$ is delayed (red):

$$\begin{array}{ccccccccc} k+i+\varsigma_{-1} & — & k+i & — & k+i+\varsigma_1 & — & k+j & — & k+j+\delta_1 \\ k+i+\varsigma_{-1} & — & k+j & — & k+i & — & k+i+\varsigma_1 & — & k+j+\delta_1 \end{array}$$

Job switching can easily be done by defining the three jobs that are affected:

1. the delayed job $(k+j)$, now after $(k+i+\varsigma_{-1})$,
2. the job performed by the YC directly after $(k+j)$, say $(k+j+\delta_1)$, now after $(k+i+\varsigma_1)$,
3. the switch job $(k+i)$ for which Eq. (H-3) holds, now after $(k+j)$,

and defining $(k+i+\varsigma_1)$ as the job performed by the YC directly following after $(k+i)$.
The matrix $\mathbf{E}$ and $\mathbf{B}$ from Eq. (5-1) should be adjusted:

$$\text{for } \mathbf{E}: \quad \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_{YC}(k+j+\delta_1) \\ \hat{x}_{YC}(k+j) \\ \hat{x}_{YC}(k+i+\varsigma_1) \\ \hat{x}_{YC}(k+i) \\ \hat{x}_{YC}(k+i+\varsigma_{-1}) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\text{for } \mathbf{E}_{switch}: \quad \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_{YC}(k+j+\delta_1) \\ \hat{x}_{YC}(k+j) \\ \hat{x}_{YC}(k+i+\varsigma_1) \\ \hat{x}_{YC}(k+i) \\ \hat{x}_{YC}(k+i+\varsigma_{-1}) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The matrix $\mathbf{B}$ follows exactly the same scheme.

The above switching methodology is focused on the QC delay, type II. Switching jobs when the YC is delayed (type I) uses the same methodology, except now the delay occurs at $(k+i)$ Furthermore, $(k+j) = (k+i+\varsigma_1)$ and $\hat{x}_{YC,ideal}(k+i)$ is described with the inequality constraints in the model equations. Hence obtaining:

$$\begin{array}{ccccccc} k+i+\varsigma_{-1} & — & k+i & — & k+j & — & k+j+\delta_1 \\ k+i+\varsigma_{-1} & — & k+j & — & k+i & — & k+j+\delta_1 \end{array}$$

while the MILP is perturbed as follows ($\mathbf{B}$ follows again the same scheme):

$$\text{for } \mathbf{E}: \quad \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_{YC}(k+j+\delta_1) \\ \hat{x}_{YC}(k+j) \\ \hat{x}_{YC}(k+i) \\ \hat{x}_{YC}(k+i+\varsigma_{-1}) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\text{for } \mathbf{E}_{switch}: \quad \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_{YC}(k+j+\delta_1) \\ \hat{x}_{YC}(k+j) \\ \hat{x}_{YC}(k+i) \\ \hat{x}_{YC}(k+i+\varsigma_{-1}) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Since for both delay types only YCs orders are switched, there will be no problem in the QC job sequence as it is constant, such that problems regarding (un)loading orders and QC drives are prevented.

Appendix I

Performance Index Travel Distance

Minimizing the travel distance is one of the objectives that is considered. The results are shown in the tables below. The traveled distance is indeed lower when it is minimized, while it compensates on the QC synchronization times.

test case 1

	mean	max	distance [m]
$\lambda_d = 0$	1095	1143	$\times$
$\lambda_d = 1$	1112	1165	2060
$\lambda_d = 10$	1161	1262	2020

test case 8

	mean	max	distance [m]
$\lambda_d = 0$	657	721	$\times$
$\lambda_d = 1$	693	781	1980
$\lambda_d = 10$	791	956	1740

Table I-1: The average and maximum QC synchronization times when the traveled distance of the AGVs is the objective.

Appendix J

Performance Index SMPL-MPS

The three different performance indices that were simulated for test cases 1, 2, 8, 9 and 17:
1) summation of the last QC synchronization times,
2) maximization of the last QC synchronization times and
3) a trade-off where the weight of the maximum time is $\lambda = 0.2 \times N$.

The last QC synchronization times of test case 17 are presented in Figure J-1, showing the difference between the maximization and summation objectives. The summation has higher synchronization times for some cranes than the maximum objective. On the other hand, due to the much lower synchronization times for other cranes, the average is minimal.

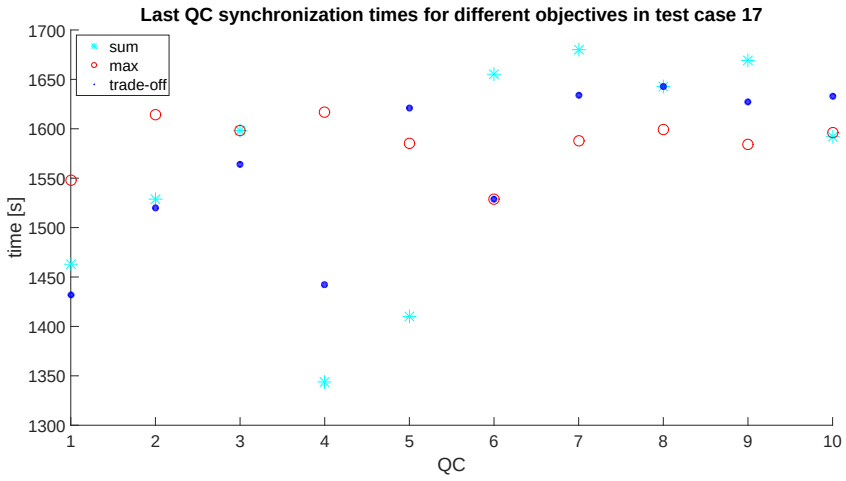

Figure J-1: Last QC synchronization times for three simulations in test case 17 using the objective functions as described above.

Especially interesting in Table J-1 is the ratio between the last synchronizing QC and the average performance. In the small cases, the amount of different routes are limited. However,

when the cases are large there are many paths through the system. The first objective (summation) has the largest deviation and lowest ratio. Only the summation is minimized, while the maximum is totally disregarded ($\omega = 1,\ \lambda = 0$ in Eq. (5-13), repeated below). The second objective (maximization) suppresses the maximum synchronization time, but compensates on others. It therefore has the largest ratio and the smallest differences in synchronization times. Now $\omega << 1$ and $\lambda = 1$ in Eq. (5-13), minimizing the maximum synchronization time. The trade-off is in the middle of these two discussed objectives, as is expected. In the equation, it shows that both the average performance as well as the maximum are minimized, since $\omega_n = 1$ and $\lambda = 0.2 \times N$.

$$\min_{x,z,t_n} J = \min_{x,z,t_n} \left(\omega_n \sum_{n=1}^{N} \hat{x}_n(k+N_p) + \lambda \max_{n\in(1,\dots,N)} \hat{x}_n(k+N_p) \right)$$

test case 1

	QC_1	QC_2	ratio
sum	5762	5808	0.996
max	5768	5783	0.999
trade-off	5768	5783	0.999

test case 2

	QC_1	QC_2	ratio
sum	4590	4500	0.980
max	4590	4500	0.980
trade-off	4590	4500	0.980

test case 8

	QC_1	QC_2	QC_3	QC_4	QC_5	ratio
sum	2396	2409	2438	2353	2432	0.987
max	2386	2404	2438	2404	2381	0.985
trade-off	2391	2409	2438	2353	2432	0.986

test case 9

	QC_1	QC_2	QC_3	QC_4	QC_5	QC_6	ratio
sum	2092	2214	2353	2277	2330	2317	0.962
max	2200	2314	2305	2299	2247	2312	0.985
trade-off	2101	2318	2249	2296	2332	2290	0.971

test case 17

	QC_1	QC_2	QC_3	QC_4	QC_5	QC_6	QC_7	QC_8	QC_9	QC_{10}	ratio
sum	1463	1529	1598	1344	1410	1655	1680	1643	1669	1592	0.928
max	1548	1614	1598	1617	1585	1529	1588	1599	1584	1596	0.981
trade-off	1432	1520	1564	1442	1621	1529	1634	1643	1627	1633	0.952

Table J-1: Varying the performance index for the test cases 1, 2, 8, 9 and 17.

Appendix K

Results of the SSMPL approach

In Chapter 7 the objective of the SSMPL model is studied and the model is validated. Additions to the results provided in the chapter can be found here.

K-1 Additional results on the performance index analysis

The average synchronization times of test case 17 can differ without influencing the objective function. This is due to the formulation of the performance index, where the average synchronization per realization is not defined. The differences in average QC performance per realization can be seen in Figure K-1.

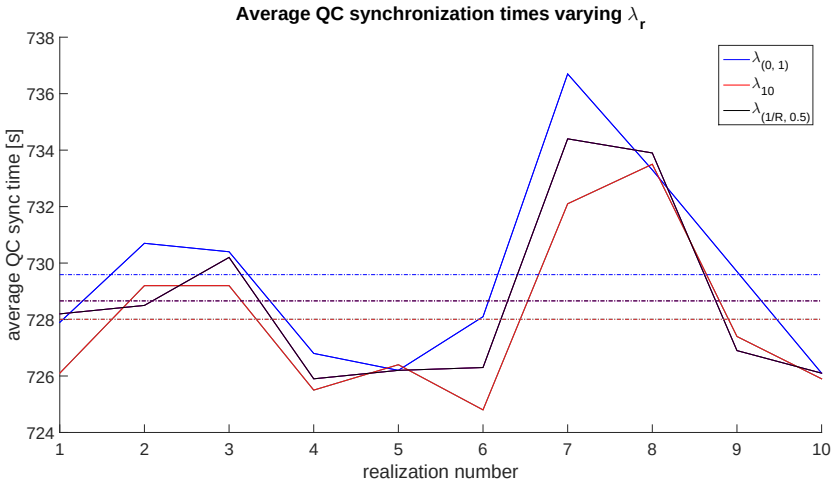

Figure K-1: When the performance weight λ_r is varied, the average QC synchronization times are different in test case 17.

K-2 Additional results on the model validation

The SSMPL model is validated by calculating a schedule when optimal, nominal and distributed values are assumed. It is claimed that the results for test case 17 show the same trend as the results for test case 1 and 8. This is supported by Table K-1 (Page 123), which shows that the SSMPL has the best performance when the 1000 "real cases" are simulated. An additional column is presented, where the distribution in the Monte Carlo algorithm is adjusted as is described in Section 7-2. The performance of this new approach is not considerably better or worse.

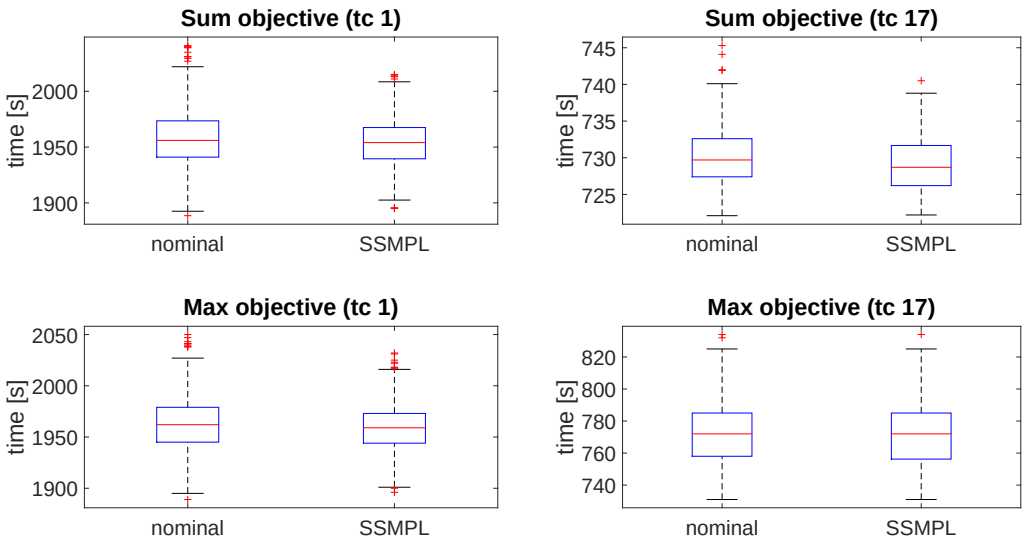

Figure K-2: Box plots of test cases 1 and 17 showing the distribution of simulating the "real cases".

Figure K-2 shows the box plots of the "real cases" of test case 1 and 17. The median is the central red mark and the box represents the value of 25% to 75% of the data. The whiskers extend to the most extreme data points (not considering outliers, which are denoted separately). The SSMPL approach seems less scattered than when nominal values are assumed, since the both the box and the whiskers cover less area of the y axis. Test case 1 shows this more clearly than test case 17. Also the whole plot of the SSMPL model appears lower than that of the nominal values, visualizing the trends found in Table K-1.

test case 17		**Optimal**	**Nominal**	**SSMPL**	**SSMPL∗**
QC_1	mean	720	720	720	723
	std	0	0	0	7
QC_2	mean	722	722	720	720
	std	3	3	0	0
QC_3	mean	732	720	720	720
	std	7	0	0	0
QC_4	mean	720	720	720	720
	std	0	0	0	0
QC_5	mean	720	720	720	720
	std	0	0	0	0
QC_6	mean	731	731	728	728
	std	19	19	17	17
QC_7	mean	771	765	771	765
	std	32	18	32	18
QC_8	mean	747	741	731	741
	std	26	26	13	26
QC_9	mean	746	730	726	732
	std	17	15	12	10
QC_{10}	mean	724	733	734	723
	std	6	7	7	6
sum	mean	7334	7302	7290	7291
	std	48	40	39	38
max	mean	783	733	772	772
	std	27	19	19	19

Table K-1: The average times and standard deviations for the QC synchronizations in test case 17. The average summation and maximum of all 1000 runs with their standard deviations are also shown.

Appendix L

Programming Importance

In the results it is shown how much time the problem formulation takes (t_b). It is interesting to note that these obtained times, of course, highly depend on the programming structure. When not enough attention is spend on the implementation, it can have a major impact on the speed of the calculations that need to be done. To illustrate the importance of the implementation, the extend of the impact on the calculation speed is explained by an example. The first time all test cases were formulated, the following code was written:

```
% Generate theta vector of MILP formulation
l_M = length(M);                                % number of ineq constr of matrix A
Mt = M(5:7);                                    % taus in column 5 to 7 of M
for ll = 1:l_M                                  % create for every constraint tau
    ii = M(ll,2);                               % AGV arrival crane
    jj = M(ll,3);                               % AGV departure crane
    tauk = [T(ii,jj);tau2(ones(n,1),:);tau3;tau5;tau6(ones(m,1),:)];
                                                % taus for only cycle k+N_max
    tau  = [0;repmat(tauk,(N_max+v),1)];        % tau for whole state
    Mt(ll,:) = tau(M(ll,5:7)+ones(1,3));        % select desired values from tau
end
theta = [Mt(:,1)+Mt(:,2)+Mt(:,3);N_max];        % add all values of tau
                                                % last value for sink node:
                                                % (N_max+v) possible sink nodes
                                                % - v occupied sink nodes = N_max
```

For every constraint that is imposed by the parameterization of the **A**-matrix a new vector θ is created. In this θ vector the value of all τ_1 depend on the travel time of this one specific constraint. When it is kept in mind that the number of constraints grow rapidly, as is seen in the results, it becomes clear that the *for*-loop runs immensely often. It even creates a new θ vector when there is no travel time in the constraint at all. The written code may be relatively short, but it describes a big effort.

To avoid any unnecessary work, another code is written that should speed up the calculation of the known θ vector in the MILP formulation. The parameterization of the **A**-matrix (the

M-matrix) indexes the positions of θ. First the indices of all τ_1 and their indices in **M** are derived. An incorrect and initial vector for θ is defined, since the traveling times are all set to zero. Then only for the constraints that contain a traveling time, the correct time is added to the initial vector θ, obtaining the correct vector.

```
% Generate theta vector of MILP formulation
tau1_pos = (0:N_max+v-1)*(3+n+m)+1;          % position of tau1 in every cycle k
M_pos = [];                                  % list to store all tau1 positions M
for ll = 1:length(tau1_pos)                  % find all positions of tau1 in M
    [pos1,pos2] = find(M(:,5)==tau1_pos(ll));
    M_pos = [M_pos;pos1];                    % store all positions
end

taus = [0;repmat([0;tau2(ones(n,1),:);tau3;tau5;tau6(ones(m,1),:)],N_max+v,1)];
                                             % create vector taus, 0 for all tau1
theta1 = taus(M(:,5)+1)+taus(M(:,6)+1)+taus(M(:,7)+1);
                                             % select desired values from taus
for ll = 1:length(M_pos)                     % for all tau1 positions
    theta1(M_pos(ll),1) = theta1(M_pos(ll),1)+T(M(M_pos(ll),2),M(M_pos(ll),3));
                                             % select 0 in taus, add the correct
                                             % traveling time from matrix T
end
theta = [theta1;N_max];
                                             % last value for sink node:
                                             % (N_max+v) possible sink nodes
                                             % - v occupied sink nodes = N_max
```

The following table shows how the formulation times of the two implementations. They both grow rapidly, but the difference is tremendously large.

nr	t_{b_1}	t_{b_2}	nr	t_{b_1}	t_{b_2}
1	0,124	0,0484	10	20,0	0,1079
2	0,253	0,0508	11	36,2	0,1292
3	0,679	0,0548	12	36,2	0,1291
4	0,947	0,0568	13	65,8	0,1541
5	2,23	0,0640	14	93,1	0,1755
6	2,89	0,0733	15	161	0,2118
7	6,00	0,0780	16	186	0,2292
8	7,37	0,0824	17	311	0,2711
9	13,7	0,0954	18	842	0,3989

Table L-1: Formulation times of the MILP problem for both implementations.

Bibliography

[1] I. H. S. Global-Insight, "Valuation of the Liner Shipping Industry - Economic Contribution and Liner Industry Operations," tech. rep., I. H. S. Global Insight, 2013.

[2] United Nations Conference on Trade and Development, "Review of maritime transport," 2013.

[3] D. Steenken, A. Voß, and R. Stahlbock, "Container terminal operation and operations research - a classification and literature review," *OR Spectrum*, vol. 26, no. 1, pp. 3–37, 2004.

[4] V. D. Nguyen and K. H. Kim, "Dispatching vehicles considering uncertain handling times at port container terminals," *Progress in Material Handling Research*, pp. 210–226, 2010.

[5] T. Ganesharajah, G. H. Nicholas, and C. Sriskandarajah, "Design and operational issues in AGV-served manufacturing systems," *Annals of Operations Research*, vol. 76, pp. 109–154, 1998.

[6] Terex Corporation, "Automated guided vehicles." `http://www.terex.com/port-solutions/en/products/new-equipment/automated-guided-vehicles/index.htm`. Accessed: 2014-09-11.

[7] B. Kersbergen, T. J. J. van den Boom, and B. De Schutter, "Reducing the time needed to solve the global rescheduling problem of trains," in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pp. 791–796, 2013.

[8] R. Goverde, "Railway timetable stability analysis using max-plus system theory," *Transportation Research Part B: Methodological*, vol. 41, pp. 179–201, 2007.

[9] R. Goverde, "A delay propagation algorithm for large-scale railway traffic networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, pp. 269–287, 2010.

[10] G. Lopes, R. Babuška, B. De Schutter, and T. van den Boom, "Switching max-plus models for legged locomotion," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO 2009)*, pp. 221–226, 2009.

[11] Y.-L. Cheng, H.-C. Sen, K. Natarajan, C.-P. Teo, and K.-C. Tan, "Dispatching automated guided vehicles in a container terminal," in *Supply Chain Optimization* (J. Geunes and P. M. Pardalos, eds.), pp. 355–389, Springer US, 2005.

[12] Bharat S Raj, "Container cranes at Kochi." `http://en.wikipedia.org/wiki/Container_crane#mediaviewer/File:Vallarpadam_Container_Terminal.JPG`. Accessed: 2014-12-12.

[13] J. Wiese, L. Suhl, and N. Kliewer, *Handbook of Terminal Planning*. Springer New York, 2011.

[14] R. Stahlbock and S. Voß, "Operations research at container terminals: a literature update," *OR Spectrum*, vol. 30, pp. 1–52, 2008.

[15] VDL, "Automatic guided vehicle AGV / container." `http://www.nauticexpo.com/prod/vdl-containersystemen/automatic-guided-vehicles-agv-container-30628-435255.html`. Accessed: 2014-12-16.

[16] L. Qui and W.-J. Hsu, "Scheduling and routing algorithms for AGVs: a survey from a computer science perspective," in *Proceedings of the 5th International Conference on Mechatronics Technology (ICMT 2001)*, pp. 112–117, 2001.

[17] C. U. Bellin, "Portalkranpaare, luftaufnahme containerlager altenwerder." `http://www.bildarchiv-hamburg.de/fotos/16/HHLA+Container+Terminal+Hamburg+Altenwerder+%28+CTA+%29/14/0150_0749+Portalkranpaare%2C+Luftaufnahme+Containerlager+Altenwerder`. Accessed: 2014-12-16.

[18] M. B. Duinkerken and J. A. Ottjes, "A simulation model for automated container terminals," in *Proceedings of Advanced Simulation Technology Conference (ASTC2000)*, 2000.

[19] TBA, "Putting AGVs to the test." `http://www.tba.nl/resources/press+section/publications/putting_{AGV}s_to_the_test.pdf`. Accessed: 2014-10-23.

[20] W. C. Ng and K. L. Mak, "Yard crane scheduling in port container terminals," *Applied Mathematical Modelling*, vol. 29, pp. 263–276, 2005.

[21] T. J. J. van den Boom and B. De Schutter, "Model predictive control of manufacturing systems with max-plus algebra," in *Formal Methods in Manufacturing* (J. Campos, C. Seatzu, and X. Xie, eds.), pp. 343–381, CRC Press, 2014.

[22] B. De Schutter and T. J. J. van den Boom, "Max-plus algebra and max-plus linear discrete event systems: An introduction," in *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES'08)*, pp. 36–42, May 2008.

[23] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Wuadat, *Synchronization and Linearity - An Algebra for Discrete Event Systems*. John Willey & Sons, New York, 1992.

[24] T. J. J. van den Boom, G. D. Lopes, and B. De Schutter, "A modeling framework for model predictive scheduling using switching max-plus linear models," in *Proceedings of Decision and Control, 2013 IEEE 52nd Annual Conference on*, pp. 5456–5461, 2013.

[25] T. J. J. van den Boom and B. De Schutter, "Modelling and control of discrete event systems using switching max-plus-linear systems," in *Proceedings of the 7th International Workshop on Discrete Event Systems (WODES'04)*, pp. 115–120, Sept. 2004.

[26] 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.* Princeton University Press, 2005.

[27] E. M. van Duinkerken, "Modelling railway infrastructure constraints in max-plus algebra," Master's thesis, Delft University of Technology - Faculty of Electrical Engineering, Mathematics and Computer Science, 2005.

[28] T. J. J. van den Boom and B. De Schutter, "Modeling and control of switching max-plus-linear systems with random and deterministic switching," *Discrete Event Dynamic Systems*, vol. 22, pp. 293–332, 2012.

[29] M. J. Gazarik and E. W. Kamen, "Reachability and observability of linear systems over max-plus," *Kybernetika*, vol. 35, pp. 2–12, 1999.

[30] F. Contu, A. Di Febbraro, and N. Sacco, "A model for performance evaluation and sensitivity analysis of seaport container terminals," in *Proceedings of the 18th IFAC World Congress*, pp. 13870–13875, 2011.

[31] F. B. van Boetzelaer, "Model predictive scheduling for container terminals," Master's thesis, Delft University of Technology - Faculty of Mechanical, Maritime and Materials Engineering, 2013.

[32] B. Kersbergen, J. Rudan, T. J. J. van den Boom, and B. De Schutter, "Towards railway traffic management using switching max-plus-linear systems." To be published, 2015.

[33] M. J. van Loenhout, "Stochastic switching max-plus-linear system - theory and aspects," Master's thesis, Delft University of Technology - Faculty of Mechanical, Maritime and Materials Engineering, 2011.

[34] T. J. J. van den Boom and B. De Schutter, "Stabilizing model predictive controllers for randomly switching max-plus-linear systems," in *Proceedings of the 2007 Control Conference (ECC07)*, pp. 4952–4959, July 2007.

[35] T. J. J. van den Boom and B. De Schutter, "Model predictive control for perturbed max-plus-linear systems: a stochastic approach," *International Journal of Control*, vol. 77, no. 3, pp. 302–309, 2004.

[36] B. De Schutter and M. Heemels, *Modeling and Control of Hybrid Systems - SC4160 course notes.* 2012.

[37] Gurobi Optimization, "Mip basics." `http://www.gurobi.com/resources/getting-started/mip-basics`. Accessed: 2015-05-25.

[38] T. J. J. van den Boom and B. De Schutter, *Optimization in Systems and Control - Lecture Notes for the Course SC4091.* 2012.

[39] J. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 3, pp. 238–252, 1962.

[40] M. Fischette, D. Salvagnin, and A. Zanette, "Minimal infeasible subsystems and benders cuts," tech. rep., DEI, University of Padova, Italy, 2008.

[41] E. Castillo, R. Minguez, A. Conejo, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming*. Springer, 2006.

[42] S. van Dijk, "Decomposition methods and rolling horizon approach for the yard crane scheduling problem," Master's thesis, Delft University of Technology - Faculty of Applied Mathematics, 2015.

[43] B. De Schutter and T. J. J. van den Boom, "Model predictive control for max-plus-linear discrete event systems," in *Automatica*, vol. 37, pp. 1049–1056, 2001.

[44] S. S. Farahani, *Approximation Methods in Stochastic Max-Plus Systems*. PhD thesis, Delft University of Technology, 2012.

[45] I. Batina, *Model predictive control for stochastic systems by randomized algorithms*. PhD thesis, Eindhoven University of Technology, 2004.

[46] J. Rudan, B. Kersbergen, T. van den Boom, and K. Hangos, "Performance analysis of MILP based model predictive control algorithms for dynamic railway scheduling," in *European Control Conference (ECC)*, pp. 4562–4567, 2013.

[47] P. Kecman, F. Corman, A. D'Ariano, and R. Goverde, "Rescheduling models for network-wide railway traffic management," *Public Transport*, vol. 5, pp. 95–123, 2013.

[48] M. Fischette, I. Ljubic, and M. Sinnl, "Thinning out facilities: a Benders decomposition approach for the incapacitated facility location problem with separable convex costs," tech. rep., DEI, University of Padova, Italy, 2015.

Glossary

List of Acronyms

AGV	automated guided vehicle
CBC	coin-or branch and cut
CDF	cumulative distribution function
CGLP	cut generation linear program
CPLEX	CPLEX
DSL	Delta Sealand
ECT	Europe Container Terminals
GLPK	GNU linear programming kit
i.i.d.	independent, identical distributed
ILP	integer linear programming
LP	linear programming
MILP	mixed-integer linear programming
MIP	mixed-integer programming
MPC	model predictive control
MPL	max-plus-linear
MPS	model predictive scheduling
PDF	probability density function
PN	Petri net
QC	quay crane

RAM random access memory

RMG rail-mounted gantry

SMPL switching max-plus-linear

SMPL-MPS switching max-plus-linear model predictive scheduling

SSMPL stochastic switching max-plus-linear

SSMPL-MPS stochastic switching max-plus-linear model predictive scheduling

SYSY ship-to-yard, ship-to-yard

SYYS ship-to-yard, yard-to-ship

S-Y ship-to-yard

TEU twenty feet equivalent unit

YC yard crane

Y-S yard-to-ship

YSSY yard-to-ship, ship-to-yard

YSYS yard-to-ship, yard-to-ship