Delft University of Technology

Delft Center for Systems and Control

Technical report 20-006

Model predictive scheduling of semi-cyclic discrete-event systems using switching max-plus linear models and dynamic graphs*

T.J.J. van den Boom, M. van den Muijsenberg, and B. De Schutter

If you want to cite this report, please use the following reference instead:

T.J.J. van den Boom, M. van den Muijsenberg, and B. De Schutter, "Model predictive scheduling of semi-cyclic discrete-event systems using switching max-plus linear models and dynamic graphs," *Discrete Event Dynamic Systems: Theory and Applications*, 2020. doi:10.1007/s10626-020-00318-w

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/20_006.html

Model predictive scheduling of semi-cyclic discrete-event systems using switching max-plus linear models and dynamic graphs

Ton J.J. van den Boom, Marenne van den Muijsenberg, and Bart De Schutter

Abstract In this paper we discuss scheduling of semi-cyclic discrete-event systems, for which the set of operations may vary over a limited set of possible sequences of operations. We introduce a unified modeling framework in which different types of semi-cyclic discrete-event systems can be described by switching max-plus linear (SMPL) models. We use a dynamic graph to visualize the evolution of an SMPL system over a certain period in a graphical way and to describe the order relations of the system events. We show that the dynamic graph can be used to analyze the structural properties of the system.

In general the model predictive scheduling design problem for SMPL systems can be recast as a mixed integer linear programming (MILP) problem. In order to reduce the number of optimization parameters we introduce a novel reparametrization of the MILP problem. This may lead to a decrease in computational complexity.

Keywords Switching max-plus linear systems · Model predictive scheduling · Mixed integer linear programming

1 INTRODUCTION

Scheduling is the process of deciding how to allocate a set of jobs to limited resources over time in such a way that one or more objectives are optimized [34, 46]. Here, a *job* is a sequence of operations according to a recipe that specifies a partial ordering among these operations, and *resources* are the equipment units where operations can take place. In a typical scheduling problem, resources are scarce and constrained in various ways (e.g., in the capacity of resources or the order of activities that can be performed on them), and one is looking for a schedule of the activities that both satisfies the constraints and is optimal

Ton J.J. van den Boom, Marenne van den Muijsenberg, and Bart De Schuttter
Delft University of Technology, Delft Center for Systems and Control, Delft, The Netherlands
Tel.: +31-15-2784052
E-mail: a.j.j.vandenboom@tudelft.nl

according to some criterion (e.g., the length of the schedule) [20]. A lot of research on scheduling has been done since the original work of Johnson [28]. Several books already presented general surveys for these problems such as [7, 12, 46] for some recent ones.

In scheduling three basic types of control decisions play a major role, namely routing, ordering, and synchronization. *Routing* decides how a job follows a sequence of resources. Usually a job follows a predetermined route through the system [28]. Sometimes there are alternative routes and the routing for each job has to be determined. Once the routes of the jobs have been fixed, conflicts may occur when multiple jobs need to be operated at the same resource. This means we have determine the *ordering* of concurring jobs in resources. Often *job synchronization* takes place, i.e. an operation of a job can only start when a specific operation of another job on another resource has finished.

A special type of scheduling problem is the flow shop scheduling problem, in which the routes of the jobs and the order of operations are fixed. All of the resources and jobs contribute to the achievement of some common goal, such as a minimal idle time and a minimal total waiting time. After all jobs in a system have been completed, the cycle is complete and a new cycle begins. An overview of the cyclic scheduling problem is available in [13, 23]. Many flow shop scheduling problems can be solved very well by considering a class of cyclic discrete event systems (DES), namely the class of max-plus linear systems [5, 25], which is capable of representing job and resource unavailability. The main reason to use these models is that the max-plus algebra is better adapted for solving sequencing problems than the classical algebra. Flow-shop scheduling problems using max-plus linear models have already been studied in [9, 18, 27, 44, 52]. Model predictive control of a flow shop scheduling problem with a just-in-time cost function and no constraint next to the non-decreasing input constraint can be solved using a semimodule approach (with a lower computational cost) [15]. More publications on control of max-plus linear system can be found in [8, 24, 26, 33, 35, 38, 40].

An example of scheduling a cyclic DES is the synchronization of the legs in a six-legged mobile robot [36]. The legs will move in a coordinated pattern, called a gait. For a fixed gait the system will show a cyclic behavior, which can be described by a max-plus linear model. Alirezaei *et al.* [2] design an optimal schedule of multiple sheets in a printer using a max-plus framework. The design variables are the feeding time and the handling time of the sheet in the duplex loop. It is shown that the resulting optimization problem can be solved using linear programming optimization methods. The scheduling of energy flows between the parallel processes in the production of calcium silicate stones is discussed in [43], where the authors use a max-plus linear model in the formulation of the scheduling problem. A railway traffic system with a cyclic timetable can described by a max-plus-linear model [11, 25]. The use of max-plus algebra allows fast computation of performance indicators and delay propagation in short time even on large networks. Modeling a cyclic DES using max-plus algebra has as advantage that the resulting max-plus-

linear (MPL) system can be considered as a linear system that has a strong analogy to conventional linear system theory.

A disadvantage of MPL systems is that the model structure is fixed, whereby changes in the structure of the system (change of route, different order of operations) cannot be modeled.

In the case of semi-cyclic processes the set of operations may vary over a limited set of possible sequences of operations. Every possible set of operations defines a mode of operation. By switching between different modes of operation (in each mode the system is described by an MPL model), we obtain a switching max-plus linear (SMPL) system [49].

An example of scheduling a semi-cyclic DES is the synchronization of the legs in a six-legged mobile robot with gait changes. The legs can change gait, which means that the legs will move in different coordinated patterns. If gait switching is possible, the system's behavior is semi-cyclic and can be described by an SMPL model [30]. Also paper handling in printers with different paper sizes and/or both simplex and duplex printing, can be modeled as an SMPL system (see Section 3). Other examples of scheduling semi-cyclic DESs are operational traffic management of railway systems with changing train orders in case of disturbances [31], and the scheduling of automated guided vehicles for unloading ships in seaport container terminals, in which the vehicles transport the containers from the quay cranes to the stack cranes [48].

Graph-based methods can be used in the scheduling procedure (see [39]). There is a close relation between max-plus linear models and the graph representation of the system [5]. From the precedence graph of the system we can compute the eigenvalue and eigenvector of the system matrix, which play a crucial role in the analysis of the system. For SMPL systems we cannot use the precedence graph because the system matrix may change every event step. In this paper we therefore use the dynamic graph [41,42]. From the dynamic graph we can compute the spectral radius of the system (equivalent to the max-plus eigenvalue for max-plus linear systems). Also controllability of the SMPL system can be studied using the dynamic graph (See Section 6).

Many scheduling problems lead to integer optimization problems, or in many cases, to mixed integer linear programming (MILP) problems [19,32,34, 46]. Also in this paper the final optimization problem will be an MILP problem. This final optimization problem will often be identical to the one we obtain using the MILP optimization problem that arises in conventional scheduling techniques. The contribution this paper is that by considering SMPL systems the performance of the scheduling procedure can be improved by using the properties of SMPL models and dynamic graphs, such as detecting bottlenecks and using reparametrization of the MILP problem based on max-plus expressions to reduce the number of optimization parameters.

Motivation and contribution

There are advantages in using SMPL systems as a basic model for scheduling. First of all there are many system-theoretical results for (S)MPL systems in

literature [5, 51]. We can use them for finding bottlenecks in the scheduling process as well as good initial scheduling values by using system properties, based on the max-plus eigenvalue and eigenvectors [30]. In this paper we discuss model predictive scheduling of semi-cyclic DESs. By using an SMPL model of the system we can make a prediction of its future behavior while searching for an optimal schedule for the future. If the model is perfect, the optimal schedule can be executed without feedback and the system will behave as predicted. However, in general the system is affected by disturbances and/or the model is not perfect, so we deal with model uncertainty. Therefore, the schedule has to be adapted on-line in response to the unexpected events. This is called operational scheduling or rescheduling.

The goal of this paper is to show how SMPL systems and their corresponding dynamic graphs can be used for the scheduling of semi-cyclic systems. We will introduce a unifying modeling approach and use it to analyze the properties of the SMPL systems. The use of a dynamic graphs eases the modeling and gives insight in the routing and order relations of the system events.

This paper extends the results of [51]: We introduce the dynamic graph for representing the switching behavior of an SMPL system. We highlight the importance of the dynamic graph concept by discussing controllability and maximum average path weight in terms of dynamic graphs and elaborate on the relation between the makespan of a schedule and the maximum average path weight of the SMPL system. Furthermore, we have added a section on classification of SMPL scheduling problems and a section with two illustrative examples of SMPL systems (with their corresponding dynamic graphs). We prove that under mild conditions the relaxed model predictive scheduling problem will give the same result as the original model predictive scheduling problem. Finally we give a generalized framework for reparametrization of the mixed integer linear programming problem and apply this to reparametrization of the ordering and routing variables.

The paper is organized as follows: In Section 2 we review SMPL systems and introduce the concept of dynamic graphs. Section 3 gives some illustrative examples of semi-cyclic systems that can be modeled as SMPL systems with their corresponding dynamics graphs. Section 4 discusses the classification of SMPL models. Section 5 analyzes the relation between the routing, ordering, and job synchronization of the scheduling operations in the context of SMPL systems. Section 6 presents some tools for the analysis of SMPL systems using dynamic graphs. Section 7 formulates the model predictive scheduling problem and shows how the related optimization problem can be solved. Section 8 concentrates on the case where the problem can be written as a mixed integer optimization problem and shows that by reparametrization the number of optimization variables can be reduced. Finally, in Section 9 some conclusions are drawn.

2 Max-plus linear systems

2.1 Max-plus algebra

Define $\varepsilon = -\infty$ and $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$. The max-plus-algebraic addition ($\oplus$) and multiplication ($\otimes$) are defined as follows [5,14]:

$$x \oplus y = \max(x, y) \quad , \quad x \otimes y = x + y$$

for any $x, y \in \mathbb{R}_\varepsilon$, and

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

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

$$[A \odot B]_{i,j} = [A]_{i,j} + [B]_{i,j}$$

for matrices $A, B \in \mathbb{R}_\varepsilon^{m \times n}$ and $C \in \mathbb{R}_\varepsilon^{n \times p}$. The last operation ($\odot$) is the max-plus Schur product. The matrix ε is the max-plus-algebraic zero matrix: $[\varepsilon]_{i,j} = \varepsilon$ for all i, j. Define for $n \in \mathbb{Z}^+$ the set $\underline{n} = \{1, 2, \ldots, n\}$. The matrix $E_n \in \mathbb{R}_\varepsilon^{n \times n}$ is the max-plus identity matrix with $[E_n]_{i,i} = 0$, $i \in \underline{n}$ and $[E_n]_{i,j} = \varepsilon$, $i \in \underline{n}$, $j \in \underline{n}$, $i \neq j$. The max-plus-algebraic matrix power of $A \in \mathbb{R}_\varepsilon^{n \times n}$ is defined as follows: $A^{\otimes^0} = E_n$ and $A^{\otimes^k} = A \otimes A^{\otimes^{(k-1)}}$ for $k = 1, 2, \ldots$.

The max-plus Kleene-star of a matrix $A \in \mathbb{R}_\varepsilon^{n \times n}$ is defined as

$$A^* = E_n \oplus A \oplus A^{\otimes^2} \oplus A^{\otimes^3} \oplus \ldots$$

Note that A^* exists for any square matrix A with a precedence graph $G(A)$ having only nonpositive circuit weights [5].

Let $u \in \mathbb{B}_\varepsilon = \{0, \varepsilon\}$ be a max-plus binary variable; then the adjoint variable $\bar{u} \in \mathbb{B}_\varepsilon$ is defined as follows:

$$\bar{u} = \begin{cases} 0 & \text{if } u = \varepsilon \\ \varepsilon & \text{if } u = 0 \end{cases}$$

2.2 Max-plus linear systems

A max-plus linear system is defined as follows [5]:

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

where $A \in \mathbb{R}_\varepsilon^{n \times n}$ and $B \in \mathbb{R}_\varepsilon^{n \times n_u}$ are the system matrices, $x(k) \in \mathbb{R}_\varepsilon^n$ is the state, $u(k) \in \mathbb{R}_\varepsilon^{n_u}$ is the input of the system, and k is the event counter of the system. In this paper k will also be called cycle counter, because in every cycle k a number of jobs are completed using a fixed route and in a specific order.

Remark 1 Note that in this paper we use the dater description of the max-plus linear system, which means that the matrices A and B are constant matrices in $\mathbb{R}_\varepsilon^{n \times n}$ and $\mathbb{R}_\varepsilon^{n \times n_u}$, respectively.

2.3 Switching max-plus linear systems

In [49] the class of switching max-plus linear (SMPL) systems was introduced, described by

$$x(k) = A(\ell(k)) \otimes x(k-1) \oplus B'(\ell(k)) \otimes u(k) \tag{2}$$

in which the matrices $A(\ell(k)) \in \mathbb{R}_{\varepsilon}^{n \times n}$, $B'(\ell(k)) \in \mathbb{R}_{\varepsilon}^{n \times n_u}$ are the system matrices for mode $\ell(k) \in \underline{n_\ell}$ for cycle k. The moments of switching between modes are determined by a switching mechanism. In general, the mode $\ell(k)$ will depend on the previous state $x(k-1)$, the previous mode $\ell(k-1)$, the input variable $u(k)$, and an additional control variable $v(k)$. For this purpose we define the switching function ϕ_{s} as:

$$\ell(k) = \phi_{\mathrm{s}}(x(k-1), \ell(k-1), u(k), v(k))$$

Model (2) is often referred to as an explicit SMPL model. In this paper, we will also consider the implicit SMPL model, given by:

$$x(k) = A_0(\ell(k)) \otimes x(k) \oplus A_1(\ell(k)) \otimes x(k-1) \oplus B(\ell(k)) \otimes u(k) \tag{3}$$

The max-plus Kleene star of $A_0(\ell(k))$ for mode $\ell(k)$ at cycle k is given by

$$[A_0(\ell(k))]^* = E_n \oplus A_0(\ell(k)) \oplus [A_0(\ell(k))]^{\otimes^2} \oplus \ldots \oplus [A_0(\ell(k))]^{\otimes^n}$$

Note that for a fixed $\ell(k)$ the matrix $A_0(\ell(k))$ is a constant matrix and so $[A_0(\ell(k))]^*$ can easily be computed. We can use the max-plus Kleene star of $A_0(\ell(k))$ to rewrite the implicit SMPL model into an explicit one. Consider an implicit SMPL system (3) for which a finite solution of $[A_0(\ell(k))]^*$ exists. Using Theorem 2.66 of [5] the implicit SMPL system can be written in the explicit form of (2), where $A(\ell(k)) = [A_0(\ell(k))]^* \otimes A_1(\ell(k))$ and $B'(\ell(k)) = [A_0(\ell(k))]^* \otimes B(\ell(k))$. In the current paper we consider the scheduling of semi-cyclic DES. We study different types of semi-cyclic DES and model them with an SMPL model. In some applications the mode $\ell(k)$ only depends on the additional integer-valued control vector $v(k)$, so $\ell(k) = \phi_{\mathrm{s}}(v(k))$. Therefore, we will also use the notation $A_0(v(k))$, $A_1(v(k))$, and $B(v(k))$.

If we rewrite the implicit SMPL model into an explicit SMPL model the system matrices of the explicit model become $A(v(k))$ and $B'(v(k))$. These matrices $A(v(k))$ and $B'(v(k))$ may become very complex functions of $v(k)$. For example, in [31] a railway traffic management problem was considered with the optimization of the scheduling parameters. A comparison was done using an explicit SMPL model and an implicit SMPL model. The explicit model became very complex and introduced many additional constraints in the final MILP problem. This made the optimization with the explicit model much slower than the optimization with the implicit model. This motivates the use of implicit models in model predictive scheduling.

Dynamic graphs

There exists a close relation between max-plus algebra and graphs [5,21,22]. Important properties such as irreducibility, eigenvalues, and structural controllability can be determined from the precedence graph of a max-plus system. If we want to study the switching behavior in the system the precedence graph cannot be used any more because the mode $\ell(k)$ and thus also the A-matrix of the system may change for every cycle k. Only if the system remains a longer time in one mode (so if $\ell(k)$ is constant in some event interval $\{k_{\text{start}}, k_{\text{start}}+1, \ldots, k_{\text{end}}\}$), we can study the behavior of the system in that specific mode by considering the precedence graph. Examples of systems that often remain a longer time in one mode for some event interval are printers, where the paper type may be constant for some time [2], and legged robots that move in a specific gait for some time [36]. For a better understanding of the switching behavior we consider the dynamic graph concept, introduced by Murota [41,42]. For the analysis of the dynamic graphs we can compute the maximum average path weight, related to the maximum growth rate of the system [50], and we can check the controllability of the system. This will be discussed in Section 6.

Consider an SMPL system of the form (3) where n_ℓ is the number of modes. For a given positive integer N, let the set $\mathcal{L}_N = \{ [\, \ell(1) \ \cdots \ \ell(N) \,]^T \mid \ell(m) \in \underline{n_\ell},\ m \in \underline{N}\}$ denote the set of all possible consecutive mode switching vectors within N cycles.

The main advantage of the dynamic graph is that it can handle the switching nature of SMPL systems. In the context of implicit SMPL systems the dynamic graph for a given mode sequence $\tilde{\ell} = [\, \ell(1) \ \cdots \ \ell(N) \,]^T \in \mathcal{L}_N$, is defined as follows:

Definition 1 Consider an implicit SMPL system for a given mode sequence $\tilde{\ell} \in \mathcal{L}_N$. The dynamic graph $G = (G_0^1, G_1^1, G_0^2, G_1^2, \ldots, G_0^m, G_1^m, H^1, \ldots, H^m)$ is a sequence of graphs, where $G_0^k = (X^k, E_0^k)$ is a directed graph with only nonpositive circuit weights, $G_1^k = (X^k, X^{k-1}, E_1^k)$, is a directed bipartite graph with E_1^k being the set of edges from X^{k-1} to X^k, and $H^k(X^k, U^k, E_{(u)}^k)$ is a directed bipartite graph with $E_{(u)}^k$ being the set of edges from U^k to X^k. The nodes X_k represent the state of a system at cycle k and the nodes U_k correspond to the input of the system at cycle k. The weight of the edge of G_0^k from node $[X^k]_j$ to $[X^k]_i$ is equal to $[A_0(\ell(k))]_{i,j}$, the weight of the edge of G_1^k from node $[X^{k-1}]_j$ to $[X^k]_i$ is equal to $[A_1(\ell(k))]_{i,j}$, and the weight of the edge of H^k from node $[U^k]_j$ to $[X^k]_i$ is equal to $[B(\ell(k))]_{i,j}$.

In the following section we will show that SMPL models and the corresponding dynamic graphs will help in the modeling of semi-cyclic DESs. System analysis with the dynamic graph will be discussed in Section 6.

Example 1 For the implicit SMPL system

$$x(k) = A_0(\ell(k)) \otimes x(k) \oplus A_1(\ell(k)) \otimes x(k-1) \oplus B(\ell(k)) \otimes u(k)$$

with two modes

$$\text{Mode 1}: \; A_0(1) = \begin{bmatrix} \varepsilon & 2 \\ \varepsilon & \varepsilon \end{bmatrix}, \; A_1(1) = \begin{bmatrix} 2 & \varepsilon \\ \varepsilon & 3 \end{bmatrix}, \; B(1) = \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 1 \end{bmatrix}$$
$$\text{Mode 2}: \; A_0(2) = \begin{bmatrix} \varepsilon & \varepsilon \\ 2 & \varepsilon \end{bmatrix}, \; A_1(2) = \begin{bmatrix} 1 & \varepsilon \\ 3 & \varepsilon \end{bmatrix}, \; B(2) = \begin{bmatrix} \varepsilon & 1 \\ \varepsilon & 1 \end{bmatrix}$$

the dynamic graph for mode-sequence $\{1, 2, 1\}$ is given in Figure 1.

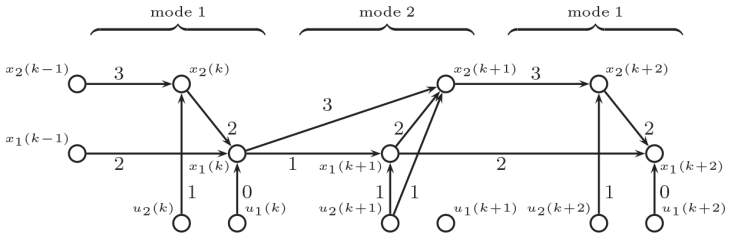

Fig. 1 Dynamic graph of the example 1.

For a periodic mode sequence with $\ell(k+c) = \ell(k)$ the dynamic graphs can be obtained from traditional precedence graphs by using a classical state-space expanding technique.

3 EXAMPLES OF SWITCHING MAX-PLUS-LINEAR SYSTEMS

In this section we discuss two applications in which the DES can be modeled as an SMPL system, namely a production system and the paper flow in a simplex/duplex printer. More examples can be found in literature (e.g. a railway network [31], a legged robot [36], and a container terminal [48]). For the mathematical modeling it is convenient if there is a general modeling methodology that one can follow. Moreover, when an adjustment must be made in the obtained model, such as adding a resource, it is desirable that not the whole system needs to be remodeled. It would be time saving if then only that specific part can be added whereby automatically a new model is obtained. Finally, a systematic way of modeling will result in a structured control problem to obtain the optimal schedule.

Due to the switching nature of SMPL systems, the Petri net representation is not useful for SMPL systems because of the changing structure. To study the mode changes in SMPL systems we consider the dynamic graph, which can represent the switching behavior of SMPL systems. In addition the dynamic graph can be used to derive the SMPL model of the system.

3.1 Production system

Consider the production system of Figure 2. This system consists of five machines $M_1, \ldots, M_5$ and operates in batches. The raw material is fed to machines M_1 and M_2 where preprocessing is done. Afterwards the intermediate product is fed to machine M_3 or machine M_4 for further processing. Finally, the two parts are assembled in machine M_5 and the end product leaves the system. We assume that each machine starts working as soon as possible on each batch, i.e., as soon as the raw material or the required intermediate product is available, and as soon as the machine is idle (i.e., the previous batch of products has been processed and has left the machine). Define $u_i(k)$, $i = 1, 2$ as the time instant at which machine M_i is fed for the kth time, $y(k)$ as the time instant at which the kth end product leaves the system, and x_i, $i = 1, \ldots, 5$ as the time instant at which machine i starts for the kth time. We define d_i, $i = 1, \ldots 5$ as the processing time on machine i for the kth batch, and we assume the transportation times between the machines can be neglected.

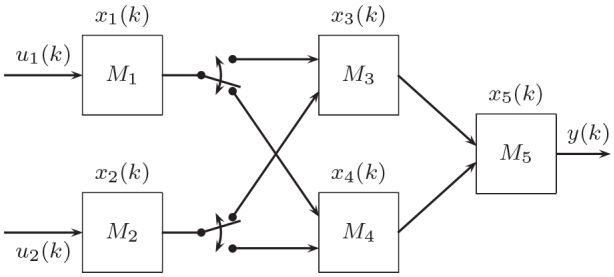

Fig. 2 Production system with five machines

Assume that the system can run in two modes. In the first mode the product from machine M_1 will proceed to machine M_3, and the product from machine M_2 will proceed to machine M_4, while in the second mode the product from machine M_1 will proceed to machine M_4, and the product from machine M_2 will proceed to machine M_3. In the first mode the system equations are given by

$$
\begin{aligned}
x_1(k) &= \max(x_1(k-1) + d_1, u_1(k)) \\
x_2(k) &= \max(x_2(k-1) + d_2, u_2(k)) \\
x_3(k) &= \max(x_1(k) + d_1, x_3(k-1) + d_3) \\
x_4(k) &= \max(x_2(k) + d_2, x_4(k-1) + d_4) \\
x_5(k) &= \max(x_3(k) + d_3, x_4(k) + d_4, x_5(k-1) + d_5) \\
y(k) &= x_5(k) + d_5
\end{aligned}
$$

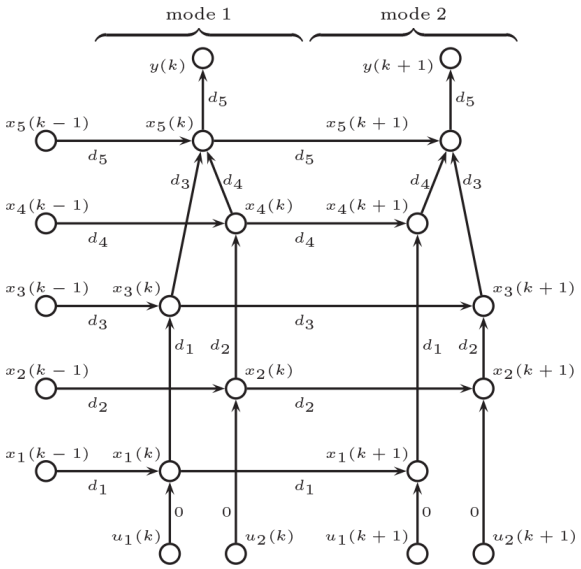

Fig. 3 Dynamic graph of the production system of Section 3.1

leading to the following matrices for the first mode:

$$A_0(1) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & d_4 & \varepsilon \end{bmatrix}, \quad A_1(1) = \begin{bmatrix} d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & d_4 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & d_5 \end{bmatrix}, \quad B(1) = \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{bmatrix}$$

In the second mode the system equations are given by

$$\begin{aligned} x_1(k) &= \max(x_1(k-1) + d_1, u_1(k)) \\ x_2(k) &= \max(x_2(k-1) + d_2, u_2(k)) \\ x_3(k) &= \max(x_2(k) + d_2, x_3(k-1) + d_3) \\ x_4(k) &= \max(x_1(k) + d_1, x_4(k-1) + d_4) \\ x_5(k) &= \max(x_3(k) + d_3, x_4(k) + d_4, x_5(k-1) + d_5) \\ y(k) &= x_5(k) + d_5 \end{aligned}$$

leading to the following matrices for the second mode:

$$A_0(2) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon \\ d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & d_4 & \varepsilon \end{bmatrix}, \quad \begin{aligned} A_1(2) &= A_1(1) \\ B(2) &= B(1) \end{aligned}$$

In Figure 3 the dynamic graph of the production system is given for the case where we have mode 1 in cycle k and mode 2 in cycle $k+1$.

3.2 Paper flow in a simplex/duplex printer

In Figure 4 a schematic representation is given of the paper path in a simplex/duplex printer (see [2]). In mode 1 (duplex mode) the paper runs from the paper input module via the image transfer station, where the image is printed onto the sheet, via the invert module and the re-entry module again to the image transfer station for back side printing. Finally, the sheet leaves the system via the paper output module. If the printer runs in mode 2 (simplex mode), the paper runs via the paper input module and image transfer station directly to the paper output module.

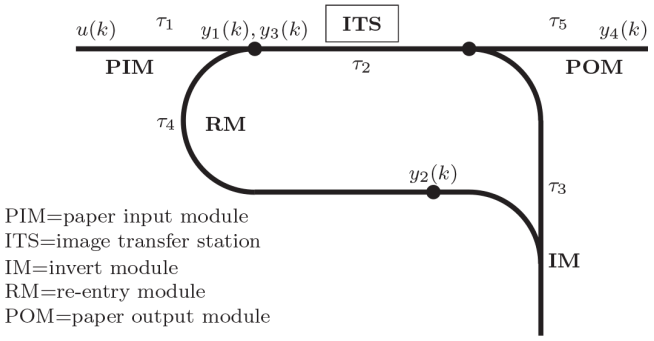

Fig. 4 Paper flow in a simplex/duplex printer

The collision avoidance of the sheets in the printer is the main constraint in the paper path. We will first derive the model and then give the dynamic graph.

Let $u(k)$ be the time instant that the kth sheet enters the PIM, $y_1(k)$ the time instant that the kth sheet enters the ITS, $y_2(k)$ the time instant that the kth sheet enters the RM, $y_3(k)$ the time instant that the kth sheet enters the ITS for the second time, $y_4(k)$ the time instant that the kth sheet leaves the printer.

The evolution of the paper path are defined as follows for the duplex mode:

$$\begin{aligned} y_1(k) &= \max(u(k) + \tau_1, y_3(k-2) + \tau_2) \\ y_2(k) &= \max(y_1(k) + \tau_2 + \tau_3, y_2(k-1) + \tau_4) \\ y_3(k) &= \max(y_1(k+1) + \tau_2, y_2(k) + \tau_4) \\ y_4(k) &= y_3(k) + \tau_2 + \tau_5 \end{aligned} \tag{4}$$

where τ_1, τ_2, τ_3, τ_4, and τ_5 are the process time for feeding, printing, handling in the first part of the loop, inverting, handling in the second part of the loop and stacking, respectively. Note that when paper sheet k is in the IM/RM, sheet $k+1$ also enters the IM/RM. This means that the second time that paper k enters the ITS ($x_3(k)$) is scheduled after the first time that paper

$k+1$ leaves the ITS $(x_1(k+1)+\tau_2)$. Therefore, the state $x_3(k)$ depends on the future event $x_1(k+1)$. In the same way, the first time paper k enters the ITS $(x_1(k))$ will be scheduled after the second time paper $k-2$ leaves the ITS $(x_3(k-2)+\tau_2)$.

To rewrite the model in the form of (3) we introduce the state x and a new input $\bar{u}$ as follows:

$$x(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k-1) \\ y_4(k-1) \end{bmatrix} \quad , \quad \bar{u}(k) = \begin{bmatrix} u(k) \\ u(k-1) \end{bmatrix}$$

The new set of state equations become

$$\begin{aligned} x_1(k) &= \max(\bar{u}_1(k)+\tau_1, x_3(k-1)+\tau_2) \\ x_2(k) &= \max(x_1(k)+\tau_2+\tau_3, x_2(k-1)+\tau_4) \\ x_3(k) &= \max(x_1(k)+\tau_2, x_2(k-1)+\tau_4) \\ x_4(k) &= x_3(k)+\tau_2+\tau_5 \end{aligned} \tag{5}$$

For the model description given in (3) we obtain the system matrices for the first mode:

$$A_0(1) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_2+\tau_3 & \varepsilon & \varepsilon & \varepsilon \\ \tau_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \tau_2+\tau_5 & \varepsilon \end{bmatrix}, \quad A_1(1) = \begin{bmatrix} \varepsilon & \varepsilon & \tau_2 & \varepsilon \\ \varepsilon & \tau_4 & \varepsilon & \varepsilon \\ \varepsilon & \tau_4 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \quad B(1) = \begin{bmatrix} \tau_1 & \varepsilon \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{bmatrix} \tag{6}$$

Now let us consider the case of mode 2 (simplex mode), where paper sheet k only needs to be printed on one side. In that case we skip the states $x_1(k)$ and $x_2(k)$ in the paper path and immediately go from the input to state $x_3(k)$. The evolution of the paper path are defined as follows for the simplex mode:

$$\begin{aligned} y_1(k) &= y_3(k-2) \\ y_2(k) &= y_2(k-1) \\ y_3(k) &= \max(y_1(k+1)+\tau_2, u(k)+\tau_1) \\ y_4(k) &= y_3(k)+\tau_2+\tau_5 \end{aligned} \tag{7}$$

With the new set $x(k)$ the state equations become

$$\begin{aligned} x_1(k) &= x_3(k-1) \\ x_2(k) &= x_2(k-1) \\ x_3(k+1) &= \max(x_1(k+1)+\tau_2, \bar{u}_2(k+1)+\tau_1) \\ x_4(k+1) &= x_3(k+1)+\tau_2+\tau_5 \end{aligned} \tag{8}$$

The dynamic graph in Figure 5 shows the paper flow for the case were sheets $k-2$, $k-1$, $k+1$, and $k+2$ are printed in duplex mode (mode 1) and sheet k is printed in simplex mode (mode 2). Note that paper sheet k will enter the system after paper sheet $k+1$. Furthermore the states $x_1(k)$ and $x_2(k)$ are actually redundant but remain in the dynamic graph to facilitate the representation of the synchronization.

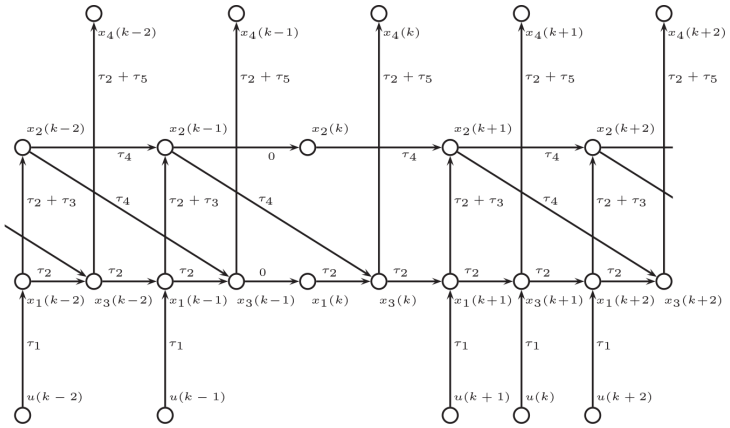

Fig. 5 Dynamic graph of the printer paper flow system of Section 3.2 in duplex mode in the cycles $k-2$, $k-1$, $k+1$, and $k+2$, but in simplex mode in cycle k.

4 CLASSIFICATION OF SWITCHING MAX-PLUS-LINEAR SYSTEMS

In the previous section we have derived SMPL models for various applications. In this section look at six features that classify the SMPL system:

1. Natural cycle.
2. Simultaneous or sequential processing.
3. Fixed or variable route.
4. Input type.
5. Due date or time table.
6. Re-entry.

These features give us information about the types of synchronization that appear in the SMPL model. We will refer to the two applications of Section 3 (production system, printer) and some SMPL systems discussed in the literature (a railway network [31], a legged robot [36], and a container terminal [48]) to illustrate these concepts.

1. Natural cycle

A natural cycle can be seen as a sequence of events whereafter the behavior of the system repeats itself. This means that after one natural cycle, all jobs enter the system again. In a railway network with a cyclic timetable, the cycle period of the timetable can be regarded as a natural cycle. For a printer the processing of one sheet of paper from the input module to the output module can be seen as a natural cycle. Similarly in a container terminal the processing of one container from quay crane to stack crane is a natural cycle. For a legged robot the natural cycle is defined as the timespan between the moment when all legs have lifted off and the moment they have all touched down again.

It is important to notice that if we choose a natural cycle it is not said that all events in a new cycle appear in the same order or appear at all: some jobs may follow another route, or the order of processing operations at the resources may be different, but the (semi-)cyclic behavior is still recognizable.

We can distinguish two cycle types for an SMPL model:

- **Batch cycle or multi-job cycle:** Every cycle consists of a batch of jobs. Sometimes the system works according to a specific periodic timetable with a fixed cycle period (railway network); sometimes multiple jobs are clearly linked by synchronization (legged robot).
- **Product cycle or single-job cycle:** The system is based on handling products in a (semi-)cyclic way (paper flow in printer, container terminal, production system). The cycle time may be different for each product. The cycle counter can be regarded as a 'product'-counter.

2. Simultaneous or sequential processing

The processing on the resources can take place in two different ways: sequentially or simultaneously. In the sequential case there is only one operation possible on a resource at the same time. The next operation on the resource has to wait until the present operation has completely finished (e.g. the printing of paper in the image transfer station has to be done sheet by sheet). In the simultaneous case, multiple operations can take place at the same resource simultaneously (e.g. multiple trains may run on the same track between stations, given a separation by the headway time). Simultaneous processing is usually due to an aggregation of smaller segments. For example most railway systems have block sections with signals to separate the trains. To analyze and control a railway network the overall model will then become too complex. In that case we aggregate a number of blocks to one block and separate trains by headway constraints.

3. Fixed or variable route

In some applications there are multiple possible routes for the jobs. However there are also applications where the routes of the jobs are fixed and only the order of the events may change (this happens in railway systems [31]). If both the routes and the order of the jobs are fixed, e.g. because operations from future cycles always come after operations from the current cycle, the SMPL system will only have one mode and we obtain an MPL system.

4. Input type

As mentioned before, $u(k)$ can take different forms. It can represent a reference under which the system needs to be operated: this is the case in a railway network where trains can never leave earlier than indicated by the reference (time table). In a printer the time instant at which paper is fed to the system can be

chosen by the user and also in the case of a production system the input can be controlled. Later on in Section 7 we use the input to control the SMPL system.

	production system	printer	railway [31]	legged robot [36]	container terminal [48]
1. multiple job cycle [M] or single job cycle [S]	S	S	M	M	S
2. simultaneous [Si] or sequential [Se]	Se	Se	Si	Se	Se
3. fixed route [F] or variable route [V]	V	V	F	F	V
4. fixed input [F], controlled input [C] or no input [-]	C	C	F	-	-
5. due date [D], time table [T] or neither [-]	D	-	T	-	-
6. re-entry [R] or no re-entry [-]	-	R	-	-	-

Table 1 Classification of the example SMPL systems

5. Due date or time table

The due date is defined as the time instant at which a job should be finished. A time table gives the desired time instant at which a job should start. Due dates and time tables are both a kind of reference signal, but they act in a different way on the SMPL system.

The entries of a time table can be seen as a lower bound on the states (e.g. in a railway system). If we consider the time table as a fixed input, the time table constraint becomes a max-plus equation on the state and this fixed input.

For some systems (e.g. the production system) there may be due dates that define when a job should be finished. The due date is a desired upper bound

on the state and may be so strict that it cannot be satisfied. The violation of the due date can then be minimized using model predictive scheduling. This will be discussed in Section 7.

6. Re-entry

A job consists of a number of operations on a sequence of resources. Usually each resource in this sequence is only visited once during the job. However, for some applications the job visits the same resource twice or more. We then talk about re-entry. In the printer example we already saw that in the duplex case the paper runs through the image transfer station twice. In the re-entry case we will then assign two states for this operation, to distinguish between the time that the first operation on that particular resource and the second operation.

Overview

In Table 1 the classification of the five examples of SMPL systems (of Section 3 and literature) is given. The six discussed features characterize the types of synchronization and as such the properties of the model. Of course for other applications different combinations of the features are possible.

5 SCHEDULING AND SMPL MODELS

In Section 3 we have reviewed some examples of applications from scheduling in which the system was described by SMPL models. In this section we will discuss how to derive an SMPL model description for scheduling in a structured and generic way.

In the previous sections we have seen that an SMPL model can switch between different modes. In some applications the number of possible mode can be very large. In that case it is often better to parameterize the modes. There are three ways to do this parametrization:

Mode parametrization: We consider the set of all possible modes and enumerate the set so that the specific mode number is the parameter.

Integer parametrization: If the set of all possible modes is large it is sometimes easier to describe the mode by a tuple of integers. The integers may describe features like the ordering of the operations for a specific resource, or determine the route for a specific job. For example in the container terminal case the mode is described by which combination quay crane, automated guided vehicle, and stack crane is used for unloading a specific container.

Binary parametrization: Binary variables can describe which of two operations go first for a resource, or can decide whether a synchronization is made or not.

Each parametrization has it advantages and disadvantages. Mode parametrization can be used if we have a small scheduling problem with a limited number of modes. For medium-size problems integer or binary parametrization will be better. If the number of possible modes is small, we can precompute the system matrices for all modes and scheduling can be done by evaluating all possible modes. When the number of modes gets large integer or binary parametrization will lead to more tractable problems. In this section we will consider binary parametrization and study the three basic types of control decisions, namely routing, synchronization, and ordering.

Routing in MPL systems

Consider a system that has to operate M jobs. For each job a specific route through the system has to be scheduled and resources have to be ordered accordingly. Let job $j \in \underline{M}$ consist of p_j operations on the resources $\mathcal{R}_j = (r_{j,1}, \ldots, r_{j,p_j})$ in processing order, and let $\mathcal{T}_j(k) = (\tau_{j,1}(k), \ldots, \tau_{j,p_j}(k))$ be the corresponding processing times in cycle k with $\tau_{j,i}(k) \geq 0$ for all i, j. Each operation is assigned to a unique machine and is not interruptible.

Finally, let $\hat{x}_j(k) = \left[x_{j,1}(k) \ldots x_{j,p_j}(k) \right]^T$ be the vector with all starting times of the operations of job j. This will give us the following inequalities for all $j \in \underline{M}$:

$$x_{j,m}(k) \geq x_{j,l}(k) + \tau_{j,l}(k), \text{ with } m > l,\ m, l \in \underline{p}_j.$$

In max-plus matrix notation this can be written as

$$\begin{bmatrix} x_{j,1}(k) \\ x_{j,2}(k) \\ \vdots \\ x_{j,p_j}(k) \end{bmatrix} \geq \begin{bmatrix} \varepsilon & \varepsilon & \ldots & \varepsilon \\ \tau_{j,1}(k) & \varepsilon & & \varepsilon \\ \vdots & \ddots & \ddots & \vdots \\ \varepsilon & \ldots & \tau_{j,p_j-1}(k) & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} x_{j,1}(k) \\ x_{j,2}(k) \\ \vdots \\ x_{j,p_j}(k) \end{bmatrix}$$

or in short notation

$$\hat{x}_j(k) \geq \hat{A}_0^{\text{job},j}(k) \otimes \hat{x}_j(k)$$

If we have M jobs, we can collect all starting times in one state vector $x(k)$ and we obtain:

$$\begin{aligned} x(k) &= \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \vdots \\ \hat{x}_M(k) \end{bmatrix} \\ &\geq \begin{bmatrix} \hat{A}_0^{\text{job},1}(k) & \varepsilon & \ldots & \varepsilon \\ \varepsilon & \hat{A}_0^{\text{job},2}(k) & & \varepsilon \\ \vdots & & \ddots & \vdots \\ \varepsilon & \ldots & \ldots & \hat{A}_0^{\text{job},M}(k) \end{bmatrix} \otimes \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \vdots \\ \hat{x}_M(k) \end{bmatrix} \\ &\geq A_0^{\text{job}}(k) \otimes x(k) \end{aligned}$$

In many applications jobs are not finished within one cycle, but need another cycle to complete. This can happen for example in a railway network with a cyclic timetable, in which a train that leaves a station in cycle $k-1$ will arrive at the next station in cycle k. The state equation is then given by

$$x(k) \geq A_0^{\text{job}}(k) \otimes x(k) \oplus A_1^{\text{job}}(k) \otimes x(k-1) \tag{9}$$

Note that in (9) we use an inequality sign instead of an equality sign. This is because the starting times may also depend on ordering and synchronization, which can delay the starting times.

Often there are alternative routes available for the jobs. Alternative routes may result in the same 'product' (e.g. various machines in production line may execute the same operation) and sometimes the route may be changed to make another 'product'.

Let there be L alternative sets of routes for this system; then for each set of routes we can define the matrices $A_{\mu,\ell}^{\text{job}}(k)$ for $\mu = 0, 1$, $\ell = 1, \ldots, n_\ell$. Let us now define a set of max-plus binary variables $(w_1(k), \ldots, w_{n_\ell}(k))$ such that if we have the ℓth set of alternative routes for the system in cycle k, we find $w_\ell(k) = 0$ and $w_j(k) = \varepsilon$ for all $j \neq \ell$. Now the job system matrices can be written for $\mu = 0, 1$ as

$$A_\mu^{\text{job}}(w(k)) = \bigoplus_{\ell=1}^{L} w_\ell \otimes A_{\mu,\ell}^{\text{job}}(k), \tag{10}$$

Ordering operations on resources in MPL systems

Consider a system with n operations, divided over N resources. Following the results of the previous paragraph, let the system allow L sets of alternative routes, parameterized by the max-plus binary variables $w(k)$. Furthermore, let $P_\ell \in {\mathbb{B}_\varepsilon}^{n \times n}$, $\ell \in \{1, \ldots, L\}$, be a matrix with max-plus binary entries, where $[P_\ell]_{i,j} = 0$ if operation i and operation j are executed on the same resource, and $[P_\ell]_{i,j} = \varepsilon$ if operation i and operation j are executed on different resources. The matrix $P(w(k))$ for assignment of the resources can now be expressed as follows:

$$P(w(k)) = \bigoplus_{\ell=1}^{L} w(k) \otimes P_\ell$$

Let $H(k)$ be a separation time matrix, where $H_{i,j}(k) \neq \varepsilon$ is the separation time between operations i and j if they may be scheduled on the same resource and $H_{i,j}(k) = \varepsilon$ if operations i and j can never be scheduled on the same resource. Finally, let $Z_\mu(k)$, $\mu = 0, 1$ be order decision matrices with max-plus binary entries, where $[Z_\mu(k)]_{i,j} = 0$ if operation i in cycle k is scheduled after operation j in cycle $k + \mu$, and $[Z_\mu(k)]_{i,j} = \varepsilon$ if operation i in cycle k is scheduled before operation j in cycle $k + \mu$. Define $z_\mu(k)$ as the vector with the stacked column vectors of matrix $Z_\mu(k)$, so $z_\mu(k) = \text{vec}(Z_\mu(k))$. Then we can use the notation $Z_\mu(k) = Z(z_\mu(k))$.

Finally define the ordering matrices

$$A_{\mu}^{\mathrm{ord}}(w(k), z_{\mu}(k)) = P(w(k)) \odot Z(z_{\mu}(k)) \odot H(k) \tag{11}$$

Now the operation ordering constraints in the system can be formulated as follows:

$$x(k) \geq A_0^{\mathrm{ord}}(w(k), z_0(k)) \otimes x(k) \oplus A_1^{\mathrm{ord}}(w(k), z_1(k)) \otimes x(k-1) \tag{12}$$

Note that certain values of $z_{\mu}(k)$ may lead to an infeasible schedule because of cycles in the ordering. An infeasible ordering occurs e.g. when in the case of three starting times of operations x_1, x_2, x_3, we choose an ordering $x_1 > x_2$, $x_2 > x_3$, and $x_3 > x_1$.

Synchronization of operations in MPL systems

Synchronization occurs when a specific operation can only start when a specific operation of another job has finished. In general we can define a number of synchronization modes $\ell = 1, \ldots, L_{\mathrm{syn}}$, where for every mode we obtain a system matrix for $\mu = 0, 1$:

$$[A_{\mu,\ell}^{\mathrm{syn}}(k)]_{ij} = \begin{cases} 0 & \text{if operation } j \text{ in cycle } k \text{ is to be scheduled behind} \\ & \text{operation } i \text{ in cycle } k-\mu. \\ \varepsilon & \text{elsewhere} \end{cases}$$

Now the operation synchronization constraints in the system can be formulated as follows:

$$x(k) \geq A_0^{\mathrm{syn}}(s_0(k)) \otimes x(k) \oplus A_1^{\mathrm{syn}}(s_1(k)) \otimes x(k-1), \tag{13}$$

where for $\mu = 0, 1$:

$$A_{\mu}^{\mathrm{syn}}(s_{\mu}(k)) = \bigoplus_{\ell=0}^{L_{\mathrm{syn}}} [s_{\mu}(k)]_{\ell} \otimes A_{\mu,\ell}^{\mathrm{syn}}(k), \tag{14}$$

where $s_{\mu}(k) \in \mathbb{B}_{\varepsilon}^{L_{\mathrm{syn}}}$ are max-plus binary variables for scheduling the synchronizations, where $[s_{\mu}(k)]_{\ell} = 0$ means that synchronization ℓ is made and $[s_{\mu}(k)]_{\ell} = 0$ means that synchronization ℓ is canceled. Synchronizations may be coupled and appear in groups (e.g. the synchronization of legs in a legged robot [30]), but can also be an isolated phenomenon (e.g. the synchronization of two trains on a platform to give passengers the chance to change trains [10]).

Combined A matrix

We have derived four conditions (9), (12), (13), and (16) for $x(k)$. We also have a set of scheduling decision variables from

- Routing: $w(k)$.
- Ordering: $z_\mu(k)$, $\mu = 0, 1$.
- Synchronization: $s_\mu(k)$, $\mu = 0, 1$.

If we now stack all decision variables into one vector

$$v(k) = \begin{bmatrix} w(k) \\ z_{\mu_0}(k) \\ z_{\mu_1}(k) \\ s_{\mu_0}(k) \\ s_{\mu_1}(k) \end{bmatrix} \in \mathbb{B}_\varepsilon^{L_{\text{tot}}}$$

where L_{tot} is the total number of scheduling variables, then we can write our scheduling model as follows

$$x(k) \geq A_0(v(k)) \otimes x(k) \oplus A_1(v(k)) \otimes x(k-1) \tag{15}$$

where for $\mu = 0, 1$:

$$\begin{aligned} A_\mu(v(k)) &= A_\mu^{\text{job}}(w(k)) \oplus A_\mu^{\text{ord}}(w(k), z_\mu(k)) \oplus A_\mu^{\text{syn}}(s_\mu(k)) \\ &= \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes A_{\mu,\ell}(k) \end{aligned}$$

Note that by choosing a specific control vector $v(k)$ the system switches between different modes of operation [49].

Reference and input signal

Some DESs work with a predefined schedule that gives a lower bound for the starting time of the operations in the system (e.g. in a railway system we have a timetable with the departure times of the trains). Let $r_i(k)$ be the starting time for operation i according to the given time schedule. To guarantees a lower bound $r_i(k)$ on operation i we introduce the constraint

$$x(k) \geq r(k). \tag{16}$$

For operations without a lower bound on the starting time we choose $r_i(k) = \varepsilon$. Some DESs have a input signal that gives the starting time of ta job (e.g. in a production system the input is the time instant at which the raw material is fed into the system). To guarantees a lower bound we introduce the constraint

$$x(k) \geq B(v(k)) \otimes u(k) \tag{17}$$

with

$$B(v(k)) = \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes B_\ell(k)$$

where $[B_\ell]_{ij}$ is the minimum time between the input event $u_j(k)$ and state event $x_i(k)$ if max-plus binary decision variable v_ℓ equals 0.

Now (15) can be replaced by

$$x(k) \geq A_0(v(k)) \otimes x(k) \oplus A_1(v(k)) \otimes x(k-1) \oplus B(v(k)) \otimes u(k) \oplus r(k) \quad (18)$$

For systems without an input signal we can discard the second term.

Final SMPL model

We now have taken into account all the constraints (9), (12), (13), (16), and (17) that determine the starting times $x(k)$. We assume that an event will take place as soon as all constraints are satisfied, which means that we now have an equality in (18) instead of an inequality:

$$x(k) = A_0(v(k)) \otimes x(k) \oplus A_1(v(k)) \otimes x(k-1) \oplus B(v(k)) \otimes u(k) \oplus r(k) \quad (19)$$

Example 2 Consider the production system from Section 3. In this system there are two routes, so we introduce the max-plus variables $w_1(k)$ and $w_2(k)$. Note that because of the fact that the system can only be in one mode at the time, we have $w_2(k) = \bar{w}_1(k)$. Now by choosing the scheduling variable $v(k) = w_1(k)$ we obtain the system matrices:

$$A_0(v(k)) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ d_1 \otimes v(k) & d_2 \otimes \bar{v}(k) & \varepsilon & \varepsilon & \varepsilon \\ d_1 \otimes \bar{v}(k) & d_2 \otimes v(k) & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & d_4 & \varepsilon \end{bmatrix}$$

$$A_1(v(k)) = \begin{bmatrix} d_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & d_4 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & d_5 \end{bmatrix}, \quad B(v(k)) = \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{bmatrix}$$

6 ANALYSIS OF SMPL SYSTEMS USING DYNAMIC GRAPHS

For max-plus linear system (1) we can determine the precedence graph $G(A)$. The maximum average circuit weight in the precedence graph $G(A)$ is then equal to the largest max-plus eigenvalue λ of the system matrix A [25].

For SMPL systems we use neither the concept of eigenvalue nor the precedence graph because the system matrix $A(\ell(k))$ may change in every cycle k.

We therefore use the concept of maximum growth rate or spectral radius. This section we start with the relation of the maximum average path weight in the dynamic graph and the maximum autonomous growth rate of the SMPL system [50]. Subsequently, we will discuss the controllability of an SMPL system in terms of the dynamic graph. This controllability is of importance when we want to perform scheduling and control of an SMPL system.

In [50] we introduced the maximum autonomous growth rate σ_{magr}:

Definition 2 [50] Consider an SMPL system of the form (2) with system matrices $A(\ell)$, $\ell \in \underline{n_\ell}$. For a given $\alpha \in \mathbb{R}$, define the matrices $A_\alpha(\ell)$ with $[A_\alpha(\ell)]_{i,j} = [A(\ell)]_{i,j} - \alpha$. Define the set $\mathcal{S}_{\text{fin},n}$ of all $n \times n$ max-plus diagonal matrices with finite diagonal entries, so $\mathcal{S}_{\text{fin},n} = \{S | S = \text{diag}_\oplus(s_1, \ldots, s_n), s_i \text{ is finite}\}$. The maximum autonomous growth rate λ of the SMPL system is defined by

$$\sigma_{\text{magr}} = \min \left\{ \alpha \;\middle|\; \exists S \in \mathcal{S}_{\text{fin},n} \text{ such that } [\, S \otimes A_\alpha(\ell) \otimes S^{\otimes^{-1}} \,]_{i,j} \leq 0, \;\; \forall\, i, j, \ell \right\}$$

Now consider the dynamic graph G of an SMPL system with mode switching vector $\tilde{\ell} = [\, \ell(1) \;\; \cdots \;\; \ell(N) \,]^T \in \mathcal{L}_N$.

If the number of cycles N runs to infinity we obtain the maximum average path weight. The maximum average path weight σ_{mapw} can be computed as follows:

$$\sigma_{\text{mapw}} = \lim_{N\to\infty} \max_{i,j\in\underline{n},\tilde{\ell}\in\mathcal{L}} 1/N \Big[A(\ell(N)) \otimes A(\ell(N-1)) \otimes \ldots \otimes A(\ell(2)) \otimes A(\ell(1)) \Big]_{ij}$$

Let S_σ be such that $[\, S_\sigma \otimes A_\alpha(\ell) \otimes {S_\sigma}^{\otimes^{-1}} \,]_{i,j} \leq 0$ and define $S_\sigma^- = {S_\sigma}^{\otimes^{-1}}$. Now we can write

$$\begin{aligned}
&\max_{i,j\in\underline{n},\tilde{\ell}\in\mathcal{L}} \Big[A(\ell(N)) \otimes \ldots \otimes A(\ell(1)) \Big]_{ij} \\
&= 1/N \max_{i,j\in\underline{n},\tilde{\ell}\in\mathcal{L}} \Big[S_\sigma^- \otimes S_\sigma \otimes A(\ell(N)) \otimes S_\sigma^- \otimes \ldots \otimes S_\sigma \otimes A(\ell(1)) \otimes S_\sigma^- \otimes S_\sigma \Big]_{ij} \\
&\leq 1/N \max_{i,j\in\underline{n},\tilde{\ell}\in\mathcal{L}} \Big[S_\sigma^- \otimes \max_{k,l\in\underline{n}} \Big[S_\sigma \otimes A(\ell(N)) \otimes S_\sigma^- \Big]_{kl} \otimes E \otimes \\
&\qquad\qquad \ldots \otimes \max_{k,l\in\underline{n}} \Big[S_\sigma \otimes A(\ell(1)) \otimes S_\sigma^- \Big]_{kl} \otimes E \otimes S_\sigma \Big]_{ij} \\
&\leq 1/N \max_{i,j\in\underline{n}} \Big[\Big[\sigma_{\text{magr}}^{\otimes^N} \otimes S_\sigma^- \otimes E \otimes S_\sigma \Big]_{ij} \Big] \\
&= \sigma_{\text{magr}} + 1/N \max_{i,j\in\underline{n}} \Big[S_\sigma^- \otimes E \otimes S_\sigma \Big]_{ij}
\end{aligned}$$

We find the bound

$$\sigma_{\text{mapw}} \leq \lim_{N\to\infty} \Big[\sigma_{\text{magr}} + 1/N \max_{i,j\in\underline{n}} \Big[S_\sigma^- \otimes E \otimes S_\sigma \Big]_{ij} \Big] = \sigma_{\text{magr}}$$

We can now summarize this in the following corollary:

Corollary 1 *Consider an SMPL system of the form (2) with system matrices* $A(\ell)$, $\ell \in \underline{n_\ell}$, *and let* λ_ℓ, $\ell \in \underline{n_\ell}$ *be the maximum eigenvalue for matrix* $A(\ell)$. *Then*

$$\max_{\ell \in \underline{n_\ell}} \lambda_\ell \leq \sigma_{\text{mapw}} \leq \sigma_{\text{magr}} \leq \max_{i,j \in \underline{n}, \ell \in \mathcal{L}} [A(\ell)]_{i,j}$$

Note the maximum average path weight is the worst case average cycle duration over the set of all possible consecutive mode switching vectors for $N \to \infty$. In optimal scheduling we optimize over all possible consecutive mode switching vectors. This means that in scheduling the maximum average path weight is an upper bound for the makespan divided by the length of the schedule.

Remark 2 The asymptotic growth rate and maximum average path weight is related to the spectral radius [45] and the Lyapunov exponent.
For a max-plus-linear system (so $n_\ell = 1$), the maximum autonomous growth rate λ is equivalent to the largest max-plus-algebraic eigenvalue of the matrix $A(1)$.

The next system property we discuss is structural controllability

Definition 3 [5,50] Consider an SMPL system of the form (2) with system matrices $A(\ell)$, $B(\ell)$, $\ell \in \underline{n_\ell}$. The SMPL system is structurally controllable if there exists a finite positive integer N such that for all mode sequences $\tilde{\ell} = \begin{bmatrix} \ell_1 \ \dots \ \ell_N \end{bmatrix}^T \in \mathcal{L}_N$ the matrices

$$\Gamma_\alpha^N(\tilde{\ell}) = \begin{bmatrix} A(\ell_N) \otimes \cdots \otimes A(\ell_2) \otimes B(\ell_1) & \dots & A(\ell_N) \otimes B(\ell_{N-1}) & B(\ell_N) \end{bmatrix}$$

are row-finite, i.e. in each row there is at least one entry different from ε.

Strong structural controllability means that all system states can be reached from the inputs for all possible mode sequences. If it is possible to choose the mode (like in case of scheduling) this concept of strong structural controllability is not always useful. We therefore also introduce the concept of weak structural controllability.

Weak structural controllability in the context of dynamic graphs is defined as follows:

Definition 4 [29] An SMPL system of the form (2) is said to be weakly structurally controllable if there exist a finite positive integer N and a mode sequence $\tilde{\ell} = \begin{bmatrix} \ell_1 \ \dots \ \ell_N \end{bmatrix}^T \in \mathcal{L}_N$ such that for any initial state $x(k)$, all state vertices at $t = k+N$ in the dynamic graph can be reached by a path originating at an input node.

Strong structural controllability (or just structural controllability) in the context of dynamic graphs is defined as follows:

Definition 5 [29] An SMPL system of the form (2) is said to be strongly structurally controllable if there exists a finite positive integer N such that the system is weakly structurally controllable for all feasible mode sequences.

This last definition corresponds to the definition of structural controllability in [50].

7 MODEL PREDICTIVE SCHEDULING

Model predictive control [16,37] is a one of the most popular advanced control strategies in industry. In this paper we will extend this methodology to scheduling and will therefore refer to it as model predictive scheduling (MPS). One of the main advantages of MPS is that we use the receding horizon principle, which means that we do not optimize the whole schedule at once, but we do this at regular time instants, where in each iteration we optimize only the jobs in the nearest future (over a certain horizon), taking into account the available information of past jobs. By using the receding horizon principle we use this available information past and only consider a limited number of future tasks. This reduces the number of scheduling variables to be optimized, which implies a lower computational burden. A too long computation time can cancel out the time gained by optimizing the schedule, or even deteriorate the total solution.

Apart from the MPC methodology to control SMPL system [49,50] one can also using residuation to obtain an appropriate schedule [3].

In this paper we aim for predictive operational scheduling, which means that based on observations of the system events we can reschedule (reroute, resynchronize, and reorder) the jobs of the system to optimize the performance. This means that in every iteration the optimization has to be done in real time using the available information of the past jobs (accumulated in the state of the SMPL system).

Consider the SMPL system of (19) and let t be the present time instant. Now we like to compute the optimal future control actions. Define the actual current cycle k as follows:

$$k = \max_{\kappa}\{\, \kappa \mid x(\kappa - 1) \leq t\} \tag{20}$$

This means that $x(\kappa) \leq t$ for $\kappa \leq k-1$, so these states are all known at time t. Note that parts of the states $x(\kappa) \leq t$ for $\kappa \geq k$ may also be known. Define the set $\mathcal{X}_t$ such that for all pairs $(i,j) \in \mathcal{X}_t$ we have $x_i(k+j) = x_i^{\text{past}}(k+j) \leq t$. Apart from the known states also specific scheduling variables and input variables are fixed at time t. Therefore, define the set $\mathcal{V}_t$ such that for all pairs $(i,j) \in \mathcal{V}_t$, $v_i(k+j) = v_i^{\text{past}}(k+j)$ is fixed at time t, and likewise define the set $\mathcal{U}_t$ such that for all pairs $(i,j) \in \mathcal{U}_t$, $u_i(k+j) = u_i^{\text{past}}(k+j) \leq t$ is fixed at time t.

The optimization problem

The MPS problem can now be formulated as finding the optimal future signal $(v(k;t), u(k;t))$, $j = 0, \ldots, N_{\text{p}} - 1$ that solves the following optimization problem for cycle k and time instant t:

$$\min_{(v(k+j;t),u(k+j;t),x(k+j;t))},\ j=0,\ldots,N_{\text{p}}-1 \quad J(k;t) \tag{21}$$

subject to

$$\begin{aligned} x(k+j;t) &= A_0(v(k+j;t);t) \otimes x(k+j;t) \\ &\quad \oplus A_1(v(k+j;t);t) \otimes x(k+j-1;t) \\ &\quad \oplus B(v(k+j;t);t) \otimes u(k+j;t) \oplus r(k+j) \end{aligned} \tag{22}$$

$$A_\mu(v(k+j;t);t) = \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes A_{\mu,\ell}(k) \;\text{ for } \mu = 0,1 \tag{23}$$

$$x_i(k+j;t) = x_i^{\text{past}}(k+j;t) \;\text{ for } (i,j) \in \mathcal{X}_t \tag{24}$$

$$u_i(k+j;t) = u_i^{\text{past}}(k+j;t) \;\text{ for } (i,j) \in \mathcal{U}_t \tag{25}$$

$$v_i(k+j;t) = v_i^{\text{past}}(k+j;t) \;\text{ for } (i,j) \in \mathcal{V}_t \tag{26}$$

where N_{p} is the prediction horizon, and $J(k;t)$ is the performance index for cycle k at time t.

Remark 3 Note that we write $x(k;t)$ rather than $x(k)$, and $A(v(k);t)$ rather than $A(v(k))$. This is due to the fact that the available information depends on time instant t, leading to a new estimate of the parameters in the system matrices, and therefore a new optimal input sequence (u, v).

The prediction horizon N_{p} determines how far (i.e. how many cycles) we look into the future to optimize the schedule. The choice of N_{p} can be seen as a compromise between global optimality and computation time. Decreasing (increasing) the prediction horizon will result in a shorter (longer) computation time and a worse (better) closed-loop performance.

The performance index $J(k;t)$ expresses a measurement of the efficiency of the chosen control vectors $v(k+j;t)$, $j = 0, \ldots, N_{\text{p}} - 1$. In some problems the makespan (i.e. the time difference between the beginning and the end of a sequence of jobs or tasks) will be the most important measure of performance (i.e. for a printer or a container terminal). The smaller the makespan, the higher the performance. For other applications (e.g. for a production system) there is a (known) due date for the finished products and the performance index will be equal to the penalty we have to pay for all delays. Also for a railway system the performance index will be equal to the penalty we have to pay for all delays with respect to a given time table (note that early departures of trains are usually not allowed). Finally, the performance index for a legged robot may vary from following a reference trajectory (with discrete targets at specific time instants) to reaching a final goal as soon as possible.

The performance index $J(k;t)$ can usually be written as

$$\begin{aligned} J(k;t) = \delta \max_{i \in \underline{n}} x_i(k+N_\mathrm{p};t) + \sum_{j=0}^{N_\mathrm{p}-1} \sum_{i=1}^{n} \kappa_{j,i}\, x_i(k+j;t) \\ + \sum_{j=0}^{N_\mathrm{p}-1} \sum_{i=1}^{n} \lambda_i \max\Big(x_i(k+j;t) - x_{\mathrm{d},i}(k+j)\,,\, 0 \Big) \\ - \sum_{j=0}^{N_\mathrm{p}-1} \sum_{m=1}^{n_u} \rho_{j,m}\, u_m(k+j;t) + \sum_{l=1}^{L_\mathrm{tot}} \sigma_{j,l}\, v_l^\flat(k+j;t). \end{aligned} \tag{27}$$

where

$$v_l^\flat(k+j;t) = \begin{cases} 0 & \text{for } v_l(k+j;t) = \varepsilon \\ 1 & \text{for } v_l(k+j;t) = 0 \end{cases} \tag{28}$$

is a conventional binary variable. Further δ, $\kappa_{j,i}$, λ_i, $\rho_{j,m}$, and $\sigma_{j,l}$ are weighting scalars, and n_u is the number of inputs of the system. The first term of (27) is related to the makespan over the prediction horizon (i.e. the total production length over the next N_p jobs), the second term is related to the weighted sum of all predicted starting times, the third term is related to the delay of the state events with respect to a specific due date signal $x_\mathrm{d}(k)$, the fourth term maximizes the inputs $u_m(k+j;t)$ (in manufacturing systems this corresponds to minimizing the number of parts in the input buffers), and the final term denotes the penalty for all changes in routing, ordering, or synchronization during cycle $k+j$.

Often we like to minimize the global makespan, i.e. the total length of the schedule. Let N_tot be the number of job cycles to be scheduled. Then the aim will be to minimize $\max_i x_i(k+N_\mathrm{tot};t)$. If N_tot is very big, it is usually better to choose a prediction horizon $N_\mathrm{p} \ll N_\mathrm{tot}$, and the criterion will be to minimize (27) where $\delta = 1$, $0 \leq \kappa_{j,i} \ll 1$, $\lambda_i = 0$ and $0 \leq \rho_{j,m} \ll 1$, and $0 \leq \sigma_{j,l} \ll 1$. A major advantage of a small prediction horizon N_p is that the computational complexity of the optimization problem is drastically reduced.

In other cases we like to minimize the sum of delays with respect to a due date signal $x_{\mathrm{d},i}(k)$ i.e. $\max\Big(x_i(k+j;t) - x_{\mathrm{d},i}(k+j)\,,\, 0\Big)$. We then have $\delta = 0$, $0 \leq \kappa_{j,i} \ll 1$, $0 \leq \lambda_i \leq 1$ and $0 \leq \rho_{j,m} \ll 1$, and $0 \leq \sigma_{j,l} \ll 1$.

The relaxed MPS problem is defined as solving (21) subject to (23)-(26), and the inequality constraints

$$\begin{aligned} x(k+j;t) \geq A_0(v(k+j;t);t) \otimes x(k+j;t) \oplus A_1(v(k+j;t);t) \otimes x(k+j-1;t) \\ \oplus B(v(k+j;t);t) \otimes u(k+j;t) \oplus r(k+j) \end{aligned} \tag{29}$$

Theorem 1 *Let $(x^\#, v^\#, u^\#)$ be an optimal solution of the relaxed model predictive scheduling problem with $J^\#$ as the optimum. If we define $x^+(k+j;t)$*

as follows:

$$x^+(k+j;t) = [A_0(v^\#(k+j;t);t)]^* \otimes \Big(A_1(v^\#(k+j;t);t) \otimes x^+(k+j-1;t) \\ \oplus B(v^\#(k+j;t);t) \otimes u^\#(k+j;t) \oplus r(k+j)\Big) \\ \text{for } j = 0,\ldots,N_\mathrm{p}-1, \tag{30}$$

and $x^+(k-1;t) = x(k-1;t)$*, then* $(x^+, v^\#, u^\#)$ *is a solution of the original model predictive scheduling problem.*

Proof:
Recall from (20) that $x(k-1;t) < t$ is completely known at time instant t.

From (30) we can derive for optimal solution $(x^\#, v^\#, u^\#)$:

$$x^\#(k+j;t) \geq [A_0(v^\#(k+j;t);t)]^* \otimes \Big(A_1(v^\#(k+j;t);t) \otimes x^+(k+j-1;t) \\ \oplus B(v^\#(k+j;t);t) \otimes u^\#(k+j;t) \oplus r(k+j)\Big) \\ \text{for } j = 0,\ldots,N_\mathrm{p}-1,$$

Now assume $x^+(k+j-1;t) \leq x^\#(k+j-1;t)$. Then we compute

$$\begin{aligned} x^\#(k+j;t) \geq & [A_0(v^\#(k+j;t);t)]^* \otimes \Big(A_1(v^\#(k+j;t);t) \otimes x^\#(k+j-1;t) \\ & \oplus B(v^\#(k+j;t);t) \otimes u^\#(k+j;t) \oplus r(k+j)\Big) \\ \geq & [A_0(v^\#(k+j;t);t)]^* \otimes \Big(A_1(v^\#(k+j;t);t) \otimes x^+(k+j-1;t) \\ & \oplus B(v^\#(k+j;t);t) \otimes u^\#(k+j;t) \oplus r(k+j)\Big) \end{aligned}$$

The term on the right-hand side of the last inequality is equal to $x^+(k+j;t)$, and so we find that if $x^+(k+j-1;t) \leq x^\#(k+j-1;t)$ then $x^+(k+j;t) \leq x^\#(k+j;t)$. For $j = 0$ we compute

$$x^+(k;t) = [A_0(v^\#(k;t);t)]^* \otimes \Big(A_1(v^\#(k;t);t) \otimes x(k-1;t) \\ \oplus B(v^\#(k;t);t) \otimes u^\#(k;t) \oplus r(k)\Big) \text{ for } j = 0,\ldots,N_\mathrm{p}-1,$$

and so $x^+(k;t) \leq x^\#(k;t)$. This means that $x^+(k+j;t) \leq x^\#(k+j;t)$ for all $j = 0,\ldots,N_\mathrm{p}-1$.

Associate $J^\#$ with the value of the cost function for the optimal solution $(x^\#, v^\#, u^\#)$ and J^+ with the value of the cost function for the solution $(x^+, v^\#, u^\#)$. Note that J is a non-decreasing function in the state x. With $x^+(k+j;t) \leq x^\#(k+j;t)$ we find $J^+ \leq J^\#$. On the other hand, note that x^+ is a feasible solution for the relaxed model predictive scheduling problem while x^* is an optimal solution. This means that $J^+ \geq J^\#$. Therefore, $J^+ = J^\#$ and $(x^+, v^\#, u^\#)$ is a solution of the original model predictive scheduling problem.

□ *End Proof*

Example 3 Consider the production system from Section 3 and Example 2, and let us introduce a signal $y_{\rm d}(k)$ which is a due date for the time instant at which output event $y(k)$ takes place. The cost criterion measures the tracking error or tardiness of the system, which is equal to the delay between the output $y(k)$ and due date $y_{\rm d}(k)$ if $y(k) - y_{\rm d}(k) > 0$, and zero otherwise. Now define $x_{\rm d}(k+j) = y_{\rm d}(k+j) - d_5$; then $y(k+j;t) - y_{\rm d}(k+j) = x_5(k+j;t) - x_{{\rm d},5}(k+j)$. Cost criterion (27) with $\delta = 0$, $\kappa_i = 0$, $\lambda_p = 0$, $p = 1, \ldots, 4$, $\lambda_5 = 1$, $0 \leq \rho_{j,l} \ll 1$, and $\sigma_{j,m} = 0$, results in

$$\begin{aligned} J(k;t) = & \sum_{j=0}^{N_{\rm p}-1} \max\Big(x_5(k+j;t) - x_{{\rm d},5}(k+j)\ ,\ 0 \Big) \\ & - \sum_{j=0}^{N_{\rm p}-1} \sum_{m=1}^{n_u} \rho_{j,m}\, u_m(k+j;t) \end{aligned} \tag{31}$$

Note that in this example we do not have a reference signal (i.e. $r(k) = \varepsilon$), so constraints (22)-(23) boil down to the set of constraints

$$\begin{aligned} x_1(k+j;t) &= \max(x_1(k+j-1;t) + d_1, u_1(k+j;t)) \\ x_2(k+j;t) &= \max(x_2(k+j-1;t) + d_2, u_2(k+j;t)) \\ x_3(k+j;t) &= \max(x_3(k+j-1;t) + d_3, x_1(k+j;t) + d_1 + \bar{v}(k+j;t), \\ &\qquad x_2(k+j;t) + d_2 + v(k+j;t)) \\ x_4(k+j;t) &= \max(x_4(k+j-1;t) + d_4, x_1(k+j;t) + d_1 + 1(k+j;t), \\ &\qquad x_2(k+j;t) + d_2 + \bar{v}(k+j;t)) \\ x_5(k+j;t) &= \max(x_5(k+j-1;t) + d_5, x_3(k+j;t) + d_3, x_4(k+j;t) + d_4) \end{aligned}$$

for $j = 1, \ldots N_p - 1$. Together with constraints (23)-(26) this gives us the MPS problem for the production system.

The mixed-integer programming problem

The MPS problem (21),(23)-(29) can be recast into a mixed-integer linear programming (MILP) problem as follows. The scheduling parameters in the MPS problem are either zero or minus infinity. For the actual numerical implementation the infinite value ε cannot be used. The first step will therefore be to replace the max-plus binary variables by conventional binary variables. We use the following approximation

$$v_i(k;t) \approx \beta\,(1 - v_i^\flat(k;t))$$

where $\beta \ll 0$ is a very large (in absolute value) negative number and $v^\flat(k;t) \in \{0,1\}^{L_{\rm tot}}$ is a conventional binary vector. The adjoint of $v_i(k;t)$ can be approximated by

$$\bar{v}_i(k;t) = \beta\, v_i^\flat(k;t)$$

Let $x^{\max} \geq \max_{i,j} x_i(k+j)$, then for $\beta \leq -x^{\max}$ the solution using the approximation and the original optimization variables will lead to the same optimal solution.

Now consider constraint (29). This can be written as a set of constraints:

$$[x(k+j;t)]_i \geq [A_0(\beta(1-v^\flat(k+j;t));t)]_{il} + [x(k+j;t)]_l \tag{32}$$

$$[x(k+j;t)]_i \geq [A_1(\beta(1-v^\flat(k+j;t));t)]_{il} + [x(k+j-1;t)]_l \tag{33}$$

$$[x(k+j;t)]_i \geq [r(k+j)]_i \tag{34}$$

for $i \in \underline{n}$, $l \in \underline{n}$, $j = 0, \ldots, N_\mathrm{p} - 1$. In the absence of a reference signal, we can drop constraint (34).

Consider cost criterion (27). Note that the first term and the second term are linear in $\tilde{x}(k;t)$, the fourth term is linear in $\tilde{u}(k;t)$, and the fifth term is linear in $\tilde{v}^\flat(k;t)$. For the third term we introduce an auxiliary variable $e(k;t) = \max(x(k;t) - x_\mathrm{d}(k), 0)$; then minimizing the third term is equal to minimizing $\sum_j \sum_i e_i(k+j;t)$, subject to

$$\begin{aligned} e_i(k+j;t) &\geq x(k+j;t) - x_\mathrm{d}(k+j) \\ e_i(k+j;t) &\geq 0 \end{aligned} \tag{35}$$

for $i \in \underline{n}$, $j = 0, \ldots, N_\mathrm{p} - 1$.

Define the vectors

$$\tilde{x}(k;t) = \begin{bmatrix} x(k;t) \\ \vdots \\ x(k+N_\mathrm{p}-1;t) \end{bmatrix}, \quad \tilde{x}_\mathrm{d}(k;t) = \begin{bmatrix} x_\mathrm{d}(k;t) \\ \vdots \\ x_\mathrm{d}(k+N_\mathrm{p}-1;t) \end{bmatrix}$$

$$\tilde{u}(k;t) = \begin{bmatrix} u(k;t) \\ \vdots \\ u(k+N_\mathrm{p}-1;t) \end{bmatrix}, \quad \tilde{v}^\flat(k;t) = \begin{bmatrix} v^\flat(k;t) \\ \vdots \\ v^\flat(k+N_\mathrm{p}-1;t) \end{bmatrix}$$

$$\tilde{e}(k;t) = \begin{bmatrix} e(k;t) \\ \vdots \\ e(k+N_\mathrm{p}-1;t) \end{bmatrix}, \quad \tilde{r}(k) = \begin{bmatrix} r(k) \\ \vdots \\ r(k+N_\mathrm{p}-1) \end{bmatrix}$$

Now cost criterion (21) can be rewritten as a linear cost criterion

$$J_\mathrm{MILP}(k) = \bar{c}_x^T\, \tilde{x}(k) + \bar{c}_v^T\, \tilde{v}^\flat(k) + \bar{c}_e^T\, \tilde{e}(k) + \bar{c}_u^T\, \tilde{u}(k) \tag{36}$$

and the MPS problem can be recast as the following MILP problem:

$$\min_{\tilde{x},\tilde{e},\tilde{v}^\flat,\tilde{u}} J_\mathrm{MILP}(k) \tag{37}$$

subject to

$$\bar{A}_x\, \tilde{x}(k) + \bar{A}_v\, \tilde{v}^\flat(k) + \bar{A}_e\, \tilde{e}(k) + \bar{A}_u\, \tilde{u}(k) \leq \bar{b}(k) \tag{38}$$

where

$$\bar{b}(k) = -\bar{A}_d\,\tilde{x}_{\mathrm{d}}(k) - \bar{A}_r\,\tilde{r}(k) - \bar{A}_{x,\mathrm{past}}\,x(k-1) + \bar{b}_0(k) \tag{39}$$

In general, MILP problems are NP-hard [47]. In practice, this means that the worst case computational complexity of a mixed integer linear programming problem grows exponentially with the number of integer values in the problem. Nevertheless, there exist fast and reliable solvers (e.g. CPLEX, Gurobi, XPRESS [4]) for these problems. Furthermore, the underlying scheduling problem may be not be NP-hard, and may be polynomially solvable (See also Section 9).

Example 4 Consider the production system from Section 3 and Examples 2 and 3.

The cost criterion measures the tracking error or tardiness of the system, which is equal to the delay between the output $y(k)$ and due date $y_{\mathrm{d}}(k)$ if $y(k)-y_{\mathrm{d}}(k) > 0$, and zero otherwise. Define $e(k+j;t) = y(k+j;t)-y_{\mathrm{d}}(k+j) = x_5(k+j;t)+d_5-y_{\mathrm{d}}(k+j)$. Now define $x_{\mathrm{d}}(k+j) = y_{\mathrm{d}}(k+j)-d_5$; then using cost criterion (27) with $\delta = 0$, $\kappa_i = 0$, $\lambda_p = 0$, $p = 1,\ldots,4$, $\lambda_5 = 1$, $0 \le \rho_{j,l} \ll 1$, and $\sigma_{j,m} = 0$, results in

$$J(k;t) = \sum_{j=0}^{N_{\mathrm{p}}-1} e(k+j;t) - \sum_{j=0}^{N_{\mathrm{p}}-1}\sum_{m=1}^{n_u} \rho_{j,m}\,u_m(k+j;t) \tag{40}$$

with constraints

$$\begin{aligned} e(k+j;t) &\ge x_5(k+j;t) + d_5 - y_{\mathrm{d},5}(k+j) \\ e(k+j;t) &\ge 0 \end{aligned}$$

for $j = 1,\ldots N_{\mathrm{p}}-1$. Note that cost criterion (40) is of the linear form (36). The constraints can be rewritten in the linear form (38)–(39). This means that the MPS problem for the production system can be recast as an MILP problem.

8 REPARAMETERIZING THE MILP

In this section we will discuss the reparametrization of the binary optimization variables of the MILP. The goal is to reduce the number of binary variables in the MILP. The complexity in MILP depends on, among other things, the number of binary variables and the number of constraints. Based on this exponential behavior, the number of binary variables in a MILP is often used as an indicator of the computational difficulty. If we can reduce the number of binary parameters without adding too many constraints, the computation of the MILP may well be reduced. However, the final computational benefits depends on how the MILP optimization problem (21)–(21) is solved, and may depend on the choice of branch and bound algorithm [6,53]. We start with the general idea of reparametrization and then look at two specific cases, namely routing and ordering.

8.1 Reparametrization with max-plus functions

We start with the definition of a max-plus function:

Definition 6 A scalar max-plus function $f : \mathbb{R}_\varepsilon^n \to \mathbb{R}_\varepsilon$ is defined by the recursive grammar

$$f(x) := x_i \mid \max(f_k(x), f_l(x)) \mid f_k(x) + f_l(x) \mid \alpha \ , \tag{41}$$

with $i \in \{1, \ldots, n\}$, $\alpha \in \mathbb{R}_\varepsilon$, and where $f_k : \mathbb{R}_\varepsilon^n \to \mathbb{R}_\varepsilon$, $f_l : \mathbb{R}_\varepsilon^n \to \mathbb{R}_\varepsilon$ are again scalar max-plus functions; the symbol $\mid$ stands for "or", and max is performed entrywise.

A vector function $f : \mathbb{R}_\varepsilon^n \to \mathbb{R}_\varepsilon^m$ is an max-plus function if all entries are scalar max-plus functions. A max-plus linear function is an max-plus function. Furthermore from the above definition it is immediately clear that the max-plus product of two max-plus functions is again an max-plus function.

Theorem 2 *A scalar max-plus function $f : \mathbb{R}_\varepsilon^n \to \mathbb{R}_\varepsilon$ can be rewritten in a canonical form*

$$f = \max_{i=1,\ldots,L} (\Gamma_i x + \delta_i)$$

for some integer L, matrix Γ and vector δ, and where Γ_i stand for the ith row of Γ.

Proof:
This theorem is a special case of Theorem 3.1 in [17].

□ *End Proof*

Introduce a max-plus function $f : \mathbb{B}_\varepsilon^{L_{\mathrm{red}}} \to \mathbb{B}_\varepsilon^{L_{\mathrm{tot}}}$ with $L_{\mathrm{red}} < L_{\mathrm{tot}}$ and let

$$v(k+j;t) = f(\theta(k+j;t)) \tag{42}$$

Now we substitute (42) in constraint (29). We obtain:

$$\begin{aligned} x(k+j;t) \geq \\ & A_0(f(\theta(k+j;t));t) \otimes x(k+j;t) \\ & \oplus A_1(f(\theta(k+j;t));t) \otimes x(k+j-1;t) \\ & \oplus B(f(\theta(k+j;t));t) \otimes u(k+j;t) \oplus r(k+j) \end{aligned} \tag{43}$$

with

$$\begin{aligned}
A_0(f(\theta(k+j;t));t) &= \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes A_{0,\ell}(k) \\
&= \bigoplus_{\ell=1}^{L_{\text{tot}}} f_\ell(\theta(k+j;t)) \otimes A_{0,\ell}(k) \\
A_1(f(\theta(k+j;t));t) &= \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes A_{1,\ell}(k) \\
&= \bigoplus_{\ell=1}^{L_{\text{tot}}} f_\ell(\theta(k+j;t)) \otimes A_{1,\ell}(k) \\
B(f(\theta(k+j;t));t) &= \bigoplus_{\ell=1}^{L_{\text{tot}}} v_\ell(k) \otimes B_\ell(k) \\
&= \bigoplus_{\ell=1}^{L_{\text{tot}}} f_\ell(\theta(k+j;t)) \otimes B_\ell(k)
\end{aligned}$$

Note that every entry of $A_\mu(f(\theta(k+j;t));t)$, $\mu = 0, 1$, and $B(f(\theta(k+j;t));t)$ is a max-plus function in the variable θ and so the right-hand side of (43) is a max-plus function in θ, x, u, and r.

Next we introduce conventional binary variables $\theta^\flat(k+j;t)$ similar to the previous section, so

$$\theta_i(k;t) \approx \beta\,(1 - \theta_i^\flat(k;t))$$

We can now find matrices $\bar{A}(k+j;t)$ such (43) can be rewritten as

$$x(k+j;t) \geq \bar{A}(k+j;t) \cdot \begin{bmatrix} x(k+j;t) \\ r(k+j) \\ \theta^\flat(k+j;t) \\ u(k+j;t) \end{bmatrix}$$

and so in the implementation constraint (43) results in a set of linear constraints.

To give an idea how to find an appropriate mapping $v(k+j;t) = f(\theta(k+j;t))$ we will discuss reparametrization of the route scheduling variables and the order scheduling variables.

8.2 Reparametrization of the routing variables

We start by looking at the routing of the jobs. From (9) and (10) we know the routing constraints can be written as

$$\begin{aligned} x(k) \geq & \bigoplus_{\ell=1}^{L} w_\ell \otimes A^{\text{job}}_{0,\ell}(k) \otimes x(k) \\ & \oplus \bigoplus_{\ell=1}^{L} w_\ell \otimes A^{\text{job}}_{1,\ell}(k) \otimes x(k-1) \end{aligned} \tag{44}$$

Now let $2^{m_0-1} < L \leq 2^{m_o}$. Then we can parameterize the variables $(w_1, \ldots, w_L)$ by m max-plus binary variables $(\eta_1, \ldots, \eta_{m_0})$ and their adjoint values $(\bar{\eta}_1, \ldots, \bar{\eta}_{m_0})$. For example if $L = 8$ we can parameterize $w_1, \ldots, w_8$ as follows:

$$\left[\begin{array}{l} w_1(k) = \bar{\eta}_1(k) \otimes \bar{\eta}_2(k) \otimes \bar{\eta}_3(k) \\ w_2(k) = \bar{\eta}_1(k) \otimes \bar{\eta}_2(k) \otimes \eta_3(k) \\ w_3(k) = \bar{\eta}_1(k) \otimes \eta_2(k) \otimes \bar{\eta}_3(k) \\ w_4(k) = \bar{\eta}_1(k) \otimes \eta_2(k) \otimes \eta_3(k) \\ w_5(k) = \eta_1(k) \otimes \bar{\eta}_2(k) \otimes \bar{\eta}_3(k) \\ w_6(k) = \eta_1(k) \otimes \bar{\eta}_2(k) \otimes \eta_3(k) \\ w_7(k) = \eta_1(k) \otimes \eta_2(k) \otimes \bar{\eta}_3(k) \\ w_8(k) = \eta_1(k) \otimes \eta_2(k) \otimes \eta_3(k) \end{array}\right. \tag{45}$$

In general we can define a max-plus function f such that $w_i(k) = f_i(\eta(k))$. By substitution of $w(k) = f(\eta(k))$ in (10) the number of variables can be reduced. Instead of $A^{\text{job}}_{\mu}(w(k))$ we will use the notation $A^{\text{job}}_{\mu}(\eta(k))$.

Note that by the reparametrization the number of scheduling variables is reduced from L to $m_0 = \lceil \log_2 L \rceil$. The difference between L and m_0 will grow rapidly with increasing L, as can be seen in Table 2.

Table 2 The number of route scheduling variables (L) and the reduced number of variables (m_0)

L	m_0
2	1
3	2
5	3

L	m_0
10	4
15	4
30	5

Remark 4 If L is not an exact power of 2, there will be more permutations of $\eta_i(k)$ and $\bar{\eta}_i(k)$ than necessary. In that case we can introduce a constraint on $\eta_i(k)$ and $\bar{\eta}_i(k)$ to describe the allowed set. Let $\eta^{\flat}_i(k)$ be the conventional binary variable associated with the max-plus binary variable $\eta_i(k)$. Then this

constraint can be defined as

$$\sum_{i=1}^{m_0} 2^{m_0-i}\,\eta^\flat{}_i \leq L-1.$$

For example, for $L=10$, and so $m_0=4$, we add the constraint:

$$8\,\eta^\flat_1 + 4\eta^\flat_2 + 2\eta^\flat_3 + \eta^\flat_4 \leq 9$$

8.3 Reparametrization of the ordering variables

Next we consider the operation ordering of the p jobs on a resource. From (12) we know that the ordering constraints can be written as

$$x(k) \geq A_0^{\mathrm{ord}}(\eta(k), z_0(k)) \otimes x(k) \oplus A_1^{\mathrm{ord}}(\eta(k), z_1(k)) \otimes x(k-1) \tag{46}$$

The vector $z_\mu(k)$, $\mu = 0, 1$ consists of the entries of matrix $Z_\mu(k)$. Note that if operation i in cycle k is scheduled after operation j, so $[Z_\mu(k)]_{i,j} = 0$, then automatically operation j in cycle k is scheduled before operation i, so $[Z_\mu(k)]_{i,j} = \varepsilon$. In general we will find that $[Z_\mu(k)]_{i,j} = \overline{[Z_\mu(k)]}_{i,j}$ for $i \neq j$ and $[Z_\mu(k)]_{i,i} = \varepsilon$. This means that for scheduling p operations the vector $z_\mu(k)$ will have dimension $m_1 = p(p-1)/2$. Note that certain values of $z_\mu(k)$ will lead to an infeasible schedule because of cycles in the ordering. To avoid these infeasible schedules we can define the set $\mathcal{Z}$ with all feasible values $z_\mu(k)$. For scheduling p operations on one resource we have $p!$ possible permutations. This is also the maximum number of elements in the set $\mathcal{Z}$. Let $m_2 = \lceil \log_2 p! \rceil$; then we need m binary parameters $\gamma_\mu(k) = \left[\, [\gamma_\mu]_1(k) \;\ldots\; [\gamma_\mu]_m(k) \,\right]^T$ to model all possible allowed values $z_\mu(k)$. We can define max-plus function $f : (\mathbb{B}_\varepsilon)^{m_1} \to (\mathbb{B}_\varepsilon)^{m_2}$ such that $[z_\mu]_i(k) = [f_\mu]_i((\gamma_\mu(k))$. A way to derive a concise description of the function f can be found in [1], where a binary decision tree is used to reduce the number of decision parameters.

Example 5 Consider an example for $p = 3$, where we have a matrix

$$Z(z_0(k)) = \begin{bmatrix} \varepsilon & \overline{[z_0]}_1 & \overline{[z_0]}_3 \\ [z_0]_1 & \varepsilon & \overline{[z_0]}_2 \\ [z_0]_3 & [z_0]_2 & \varepsilon \end{bmatrix}$$

with $p(p-1)/2 = 3$ variables. We have $p! = 6$ feasible combinations of $([z_0]_1, [z_0]_2, [z_0]_3)$ and so $m = \lceil \log_2 3! \rceil = 3$.

Table 3 shows the permutations (last column) with the corresponding values of the entries z_0 of the matrix Z_0 for the ordering $P(1) - P(2) - P(3)$. From the table we can see that

$$\begin{aligned}
[z_0]_1 &= ([\bar{\gamma}]_0]_1 \otimes [\bar{\gamma}]_0]_2) \oplus ([\gamma_0]_1 \otimes [\gamma_0]_3) \oplus ([\gamma_0]_1 \otimes [\gamma_0]_2) \\
[z_0]_2 &= [\gamma_0]_2 \oplus ([\bar{\gamma}]_0]_1 \otimes [\bar{\gamma}]_0]_3) \\
[z_0]_3 &= ([\bar{\gamma}]_0]_1 \otimes [\bar{\gamma}]_0]_2) \oplus ([\bar{\gamma}]_0]_1 \otimes [\bar{\gamma}]_0]_3) \oplus ([\gamma_0]_1 \otimes [\gamma_0]_2)
\end{aligned}$$

Table 3 Max-plus truth table and corresponding permutation

	0	1	2	3	4	5	6	7
$[\gamma_0]_1$	ε	ε	ε	ε	0	0	0	0
$[\gamma_0]_1$	ε	ε	0	0	ε	ε	0	0
$[\gamma_0]_1$	ε	0	ε	0	ε	0	ε	0
$[z_0]_1$	0	0	ε	ε	ε	0	0	0
$[z_0]_2$	0	ε	0	0	ε	ε	0	0
$[z_0]_3$	0	0	0	ε	ε	ε	0	0
$P(1)$	1	1	2	2	3	3	*	*
$P(2)$	2	3	1	3	1	2	*	*
$P(3)$	3	2	3	1	2	1	*	*

Subsequently, we can substitute $z_\mu(k) = f_\mu((\gamma_\mu(k)))$ into (11) and we obtain

$$A_\mu^{\text{ord}}(\eta(k), \gamma_\mu(k)) = P(\eta(k)) \odot Z(f(\gamma_\mu(k))) \odot H(k)$$

Using the variables $\theta(k+j;t)$ (e.g. $\eta_2(k)$ or $\gamma_\mu(k)$) has two important advantages: we need less parameters and there are less infeasible choices for $v(k)$.

Remark 5 Note that for the case in Example 5 there is no reduction because for $p = 3$ we find $m_1 = 3$ and $m_2 = 3$. For higher values of p however the difference between m_1 and m_2 grows very rapidly, as can be seen in Table 4

Table 4 The number of operations (p), the nominal number of ordering scheduling variables (m_1), and the reduced number of variables (m_2)

p	m_1	m_2
2	1	1
3	3	3
5	10	7
10	45	22
15	105	41
30	435	108

8.4 Parameter reduction

We may conclude that for every job the reduction of the number of routing variables is $\lceil \log_2 L \rceil$ where L is the number of routes for that particular job and that for every resource the reduction of routing variables is $(\lceil \log_2 p! \rceil)/p(p-1)$ where p is the number of jobs for that particular resource. Note that in both cases the number of binary variables decreases in a logarithmic way. This is often beneficial for the optimization.

9 Conclusions

In this paper we have shown that various semi-cyclic discrete-event systems can be modeled as switching max-plus linear (SMPL) systems and that dynamic graphs are a useful tool in the modeling and analysis process. The SMPL model can be applied in the context of scheduling. This can be done by formulating the model predictive scheduling problem, which can be recast as a mixed-integer linear program. We have shown how the number of optimization parameters can be reduced in a logarithmic way, which is often beneficial for the optimization.

As was already stated in Section 7, MILP problems are NP-hard [47]. However the underlying scheduling problem is not always NP-hard, and may be polynomially solvable. In future research we will continue our study on the computational complexity of the scheduling problem defined in this paper. Further we will do an in-depth investigation on how to accurately characterize the effect of the reparametrization of Section 8 on the computation time.

Acknowledgments

The authors would like to thank Vangelis Kalamboukis and Jacob van der Woude of Delft University of Technology for the inspiring discussions on dynamic graphs, and also the anonymous referees for their helpful comments on an earlier draft of this paper.

References

1. S.B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C–27(6):509–516, 1978.
2. M. Alirezaei, T.J.J. van den Boom, and R. Babuška. Max-plus algebra for optimal scheduling of multiple sheets in a printer. In *Proceedings of the American Control Conference 2012*, pages 1973–1978, Montreal, Canada, June 2012.
3. M. Alsaba, S. Lahaye, and J-L. Boimond. On just in time control of switching max-plus linear systems. In *Proceedings of ICINCO 06*, pages 79–84, Setubal, Portugal, 2006.
4. A. Atamtürk and M.W.P. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, November 2005.
5. F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Synchronization and Linearity*. Wiley, New York, 1992.
6. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35: 407–427, 1999.
7. J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling Computer and Manufacturing Processes, second ed.* Springer, Berlin, 2001.
8. J.L. Boimond and J.L. Ferrier. Internal model control and max-algebra: Controller design. *IEEE Transactions on Automatic Control*, 41(3):457–461, March 1996.
9. J.L. Bouquard, C. Lenté, and J.C. Billaut. Application of an optimization problem in max-plus algebra to scheduling problems. *Discrete Applied Mathematics*, 154(15):2064–2079, 2006.
10. J.G. Braker. Max-algebra modelling and analysis of time-table dependent transportation networks. In *Proceedings of the 1st European Control Conference*, pages 1831–1836, 1991.

11. J.G. Braker. Algorithms and applications in timed discrete event systems. PhD thesis, Delft University of Technology, Delft, 1993.
12. P. Brucker. *Scheduling Algorithms, third ed.* Springer, Berlin, 2001.
13. P. Brucker and T. Kampmeyer. A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics*, 156(13):2561–2572, 2008.
14. R.A. Cuninghame-Green. *Minimax Algebra, volume 166 Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1979.
15. G.G̃omes da Silva and C.A. Maia. On just-in-time control of timed event graphs with input constraints: a semimodule approach. *Discrete Event Dynamic Systems*, 26(2):351–366, 2016.
16. B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.
17. B. De Schutter and T.J.J. van den Boom. MPC for continuous piecewise-affine systems. *System and Control Letters*, 52(3/4):179–192, 2004.
18. A. Doustmohammadi and E.W. Kamen. Direct generation of event-timing equations for generalized flow shop systems. In *Proceedings of the Photonics East'95: International Society for Optics and Photonics*, pages 50–62, 1995.
19. C.A. Floudas and X. Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139:131–162, 2005.
20. M.P.J. Fromherz. Constraint-based scheduling. In *Proceedings of the American Control Conference (ACC'01), Invited tutorial paper*, 3231–3244, June 2001.
21. M. Gondran and M. Minoux. Eigenvalues and eigenvectors in semimodules and their interpretation in graph theory. In *Proceedings of the 9th International Mathematical Programming Symposium*, 333–348, 1976.
22. M. Gondran and M. Minoux. *Graphs and Algorithms*. John Wiley & Sons, Chichester, UK, 1984.
23. C. Hanen and A. Munier. Cyclic scheduling on parallel processors: An overview. In P. Chretienne, E.G. Coffman, J.K. Lenstra, and Z. Liu, editors, *Scheduling Theory and Its Applications*, chapter 9. John Wiley & Sons Ltd, New York, 1995.
24. L. Hardouin, B. Cottenceau, M. Lhommeau, and E. Le Corronc. Interval systems over idempotent semiring. *Linear Algebra and Its Applications*, 431(5-7):855–862, 2009.
25. B. Heidergott, G.J. Olsder, and J. van der Woude. *Max Plus at Work: Modeling and Analysis of Synchronized Systems*. Princeton University Press, Princeton, 2006.
26. L. Houssin, S. Lahaye, and J.-L. Boimond. Control of (max,+)-linear systems minimizing delays. *Discrete Event Dynamic Systems*, 23(3):261–276, 2013.
27. D. Indriyani and Subiono. Scheduling of the crystal sugar production system in sugar factory using max-plus algebra. *International Journal of Computing Science and Applied Mathematics*, 2(3):33–37, 2016.
28. S.M. Johnson. Optimal two- and three-stage production schedule with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.
29. V.P. Kalamboukis. Matrix spans in max-plus algebra and a graph-theoretic approach to switching max-plus linear systems. MSc thesis Delft, University of Technology, 2018.
30. B. Kersbergen, G.A.D. Lopes, T. van den Boom, B. De Schutter, and R. Babuška. Optimal gait switching for legged locomotion. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS-2011)*, pages 2729–2734, San Francisco, USA, September 2011.
31. B. Kersbergen, J. Rudan, T. van den Boom, and B. De Schutter. Towards railway traffic management using switching max-plus-linear systems - structure analysis and rescheduling. *Discrete Event Dynamic Systems: Theory and Applications*, 26(2):183–223, 2016.
32. H. Ku and I.A. Karimi. Scheduling in serial multiproduct batch processes with finite interstage storage: A mixed integer linear programming formulation. *Industrial & Engineering Chemistry Research*, 27:1840–1848, 1988.
33. S. Lahaye, J.-L. Boimond, and J.-L. Ferrier. Just-in-time control of time-varying discrete event dynamic systems in (max,+) algebra. *International Journal of Production Research*, 46(19):5337–5348, October 2008.
34. J.Y-T. Leung, editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc, Boca Raton, FL, USA, 2004.

35. L. Libeaut and J.J. Loiseau. Admissible initial conditions and control of timed event graphs. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2011–2016, New Orleans, Louisiana, December 1995.
36. G. Lopes, B. Kersbergen, T.J.J. van den Boom, B.De Schutter, and R. Babuška. Modeling and control of legged locomotion via switching max-plus models. *IEEE Transactions on Robotics*, 30(3):652–665, 2014.
37. J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Pearson Education Limited, Harlow, UK, 2002.
38. C.A. Maia, L. Hardouin, R. Santos-Mendes, and B. Cottenceau. Optimal closed-loop control of timed event graphs in dioids. *IEEE Transactions on Automatic Control*, 48(12):2284–2287, December 2003.
39. A. Mascis and D. Pacciarelli. Job shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research*, 143(3):498–517, 2002.
40. E. Menguy, J. Boimond, L. Hardouin, and J. Ferrier. Just-in-time control of timed event graphs: Update of reference input, presence of uncontrollable input. *IEEE Transactions on Automatic Control*, 45(11):2155–2158, September 2000.
41. K. Murota. *Matrices and Matroids for Systems Analysis*, volume 20. Springer Science & Business Media, 2009.
42. K. Murota. *Systems Analysis by Graphs and Matroids: Structural Solvability and Controllability*, volume 3. Springer Science & Business Media, 2012.
43. M. Mutsaers, L. Özkan, and T. Backx. Scheduling of energy flows for parallel batch processes using max-plus systems. In *Proceedings of the 8th IFAC International Symposium on Advanced Control of Chemical Processes 2012*, pages 176–181, July 2012.
44. A. Nambiar and R. Judd. Max-plus-based mathematical formulation for cyclic permutation flow-shops. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2:85–97, 2011.
45. A. Peperko. On the continuity of the generalized spectral radius in max-algebra. *Linear Algebra and its Applications*, 435:902–907, 2011.
46. M. Pinedo. *Scheduling: Theory, Algorithms and Systems, second ed.* Prentice-Hall, Englewood Cliffs, NJ, 2001.
47. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, UK, 1986.
48. F.B. van Boetzelaer, T.J.J. van den Boom, and R.R. Negenborn. Model predictive scheduling for container terminals. In *Proceedings of the 19th IFAC World Congress (IFAC WC'14)*, pages 5091–5096, Cape Town, South Africa, August 2014.
49. T.J.J. van den Boom and B. De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14(10):1199–1211, October 2006.
50. 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*, 12(3):293–332, September 2012.
51. T.J.J. van den Boom, G.A.D. Lopes, and B. De Schutter. A modeling framework for model predictive scheduling using switching max-plus linear models. In *Proceedings of the 52st IEEE Conference on Decision and Control (CDC 2013)*, Florence, Italy, December 10-13 2013.
52. M. Yurdakul and N.G. Odrey. Development of a new dioid algebraic model for manufacturing with the scheduling decision making capability. *Robotics and Autonomous Systems*, 49:207–218, 2004.
53. H.P. Williams, *Model building in mathematical programming* (3rd ed.). New York: Wiley. 1993.