Delft University of Technology

Fac. of Information Technology and Systems **Control Systems Engineering**

Technical report bds:00-20

Model predictive control for discrete-event systems with soft and hard synchronization constraints*

B. De Schutter and T. van den Boom

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

B. De Schutter and T. van den Boom, "Model predictive control for discrete-event systems with soft and hard synchronization constraints," *Proceedings of the Workshop on Max-Plus Algebras and Their Applications to Discrete-Event Systems, Theoretical Computer Science, and Optimization*, Prague, Czech Republic, 6 pp., Aug. 2001.

Control Systems Engineering
Faculty of Information Technology and Systems
Delft University of Technology
Delft, The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
Current URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/00_20.html`

MODEL PREDICTIVE CONTROL FOR DISCRETE-EVENT SYSTEMS WITH SOFT AND HARD SYNCHRONIZATION CONSTRAINTS

Bart De Schutter * **Ton van den Boom** *

* *Control Lab, Fac. ITS, Delft Univ. of Technology*
P.O. Box 5031, 2600 GA Delft, The Netherlands
{*b.deschutter,t.j.j.vandenboom*}*@its.tudelft.nl*

Abstract: Max-plus-linear models can be used to model discrete-event systems with only synchronization and no concurrency. The synchronization constraints in max-plus-linear discrete-event systems are hard, i.e., they cannot be broken under any circumstance. We consider a class of discrete-event systems with both hard and soft synchronization constraints, i.e., if necessary, some synchronization conditions may be broken, but then a penalty is incurred. We show that — after introducing control variables — this leads to a max-plus-bilinear model. Furthermore, we also show how the model predictive control (MPC) framework can be extended to this class of discrete-event systems.

Keywords: predictive control, adaptive control, discrete-event systems, operations research, manufacturing systems, railways, synchronization

1. INTRODUCTION

1.1 Overview

Max-plus-linear discrete-event systems are a class of discrete-event systems (DES) in which only synchronization and no concurrency or choice occurs. So typical examples are serial production lines, production systems with a fixed routing schedule, and railway networks with rigid connection constraints. Max-plus-linear DES can be described by a model that is "linear" in the max-plus algebra (Baccelli *et al.*, 1992; Cohen *et al.*, 1985; Cuninghame-Green, 1979).

The synchronization constraints for max-plus-linear DES are "hard", i.e., these constraints should always be met. In this paper we propose a modeling and control framework for a class of DES with both *soft* and *hard* synchronization constraints: in some cases we allow an event to start although not all scheduled predecessor events have been completed, but at a cost. This could occur in a manufacturing, planning, logistics, or railway operations context. Consider, e.g., a large manufacturing process, with a pre-scheduled planning. Suppose that in order to start a given operation, we need the intermediary output products of some predecessor operations. If the synchronization constraints are hard, an operation can only start if all predecessor operations have been completed, which might lead to a propagation of delays if one of the predecessor operations is not completed in time. Especially if the planning is tight, this could cause serious (financial) problems. Therefore, we could impose soft synchronization constraints so that we can start the operation anyway if further delay would lead to large deviations from the planned schedule. However, in that case we incur a penalty cost (e.g., if the intermediary output products of the delayed

predecessor operation have to be bought from a third party).

We also present a model predictive control (MPC) framework for DES with both hard and soft synchronization constraints. There are several reasons why MPC is the most applied advanced control technique in the process industry: MPC is a easy-to-tune model-based controller design procedure that can handle multi-input multi-output processes and constraints on the inputs and outputs of the process. Furthermore, MPC can handle structural changes, such as sensor or actuator failures and changes in the system parameters, by using a moving horizon approach, in which the model and the control strategy are continuously updated. For more information on MPC we refer to (Camacho and Bordons, 1995; Clarke *et al.*, 1987; García *et al.*, 1989).

Conventional MPC uses discrete-time models (i.e., models consisting of a system of difference equations). In (De Schutter and van den Boom, 2000) we have extended MPC to the class of max-plus-linear DES. So there we have only considered hard synchronization constraints. In this paper, we further extend the MPC framework to DES with both hard and soft synchronization constraints. The proposed MPC approach has the following ingredients (which are also present in conventional MPC): a prediction horizon, a receding horizon procedure, and a regular update of the model and re-computation of the optimal control input. In general, this leads to a hard non-convex nonlinear optimization problem. However, we show that the trajectories of a DES with hard and soft synchronization constraints can be described by an extended linear complementarity problem (ELCP), for which we can compute a parameterized solution using the algorithm described in (De Schutter and De Moor, 1995). Afterwards, we can compute the optimal control over the parameters of the resulting solution set of the ELCP. The advantage is that we now have to solve a sequence of optimization problems with a convex feasible set (although the objective function is still nonlinear and non-convex). Computational experiments show that (for small sized problems or for a small control horizon) the ELCP approach is much faster and yields a better minimum than the straightforward nonlinear optimization approach.

1.2 Notation

We assume that the reader is familiar with the basic concepts of max-plus algebra (Baccelli *et al.*, 1992; Cuninghame-Green, 1979). We define

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

for $x, y \in \mathbb{R}_\varepsilon$ with $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{-\infty\}$. For matrices $A, B \in \mathbb{R}_\varepsilon^{m\times n}$ and $C \in \mathbb{R}_\varepsilon^{n\times p}$ we have

$$\begin{aligned}(A \oplus B)_{ij} &= (A)_{ij} \oplus (B)_{ij} = \max((A)_{ij}, (B)_{ij}) \\ (A \otimes C)_{ij} &= \bigoplus_{k=1}^{n} (A)_{ik} \otimes (C)_{kj} \\ &= \max_{k=1,\ldots,n} ((A)_{ik} + (C)_{kj})\,.\end{aligned}$$

In (Baccelli *et al.*, 1992; Cuninghame-Green, 1979) it is shown that DES in which there is (hard) synchronization but no concurrency can be described by a model of the form

$$x(k) = A(k) \otimes x(k-1) \oplus B(k) \otimes u(k) \qquad (1)$$
$$y(k) = C(k) \otimes x(k)\ . \qquad (2)$$

The index k is called the event counter. The state $x(k)$ typically contains the time instants at which the internal events occur for the kth time, the input $u(k)$ contains the time instants at which the input events occur for the kth time, and the output $y(k)$ contains the time instants at which the output events occur for the kth time. For later reference, we note that (1) can be rewritten as

$$x_j(k) = \bigoplus_{i=1}^{n} (A(k))_{ji} \otimes x_i(k-1) \oplus \bigoplus_{i=1}^{n_u} (B(k))_{ji} \otimes u_i(k) \qquad (3)$$

for $j = 1, 2, \ldots, n$ where n is the system order and n_u is the number of inputs.

2. DES WITH SOFT SYNCHRONIZATION CONSTRAINTS

Consider a system in which some operations depend on some predecessor operations (cf. Figure 1). The system is allowed to operate in cycles. Let n be the number of operations and let $x_j(k)$ be the time instant at which operation j starts during the kth operation cycle. Let $d_j(k)$ be the scheduled starting time of operation j during the kth cycle according to the planning.

Let $\mathcal{C}_j(k)$ be the set of predecessor operations for operation j in cycle k. This set can be divided in a set $\mathcal{C}_j^{\text{hard}}(k)$ of predecessor operations with hard synchronization conditions (i.e., operation j can only be started if these predecessor operations have been finished, no matter how much time they

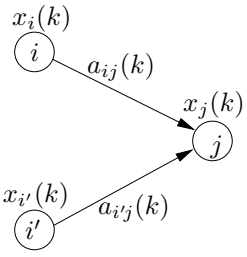

Fig. 1. An operation and its predecessors.

are delayed), and a set $\mathcal{C}_j^{\text{soft}}(k)$ of predecessor operations with soft synchronization conditions (i.e., operation j can be started before these predecessor operations have been finished, but then a penalty has to be paid). Let $c_{ij}^{\text{broken}}(k)$ be the (maximal) cost associated with a broken synchronization (see (6) for a refinement)). We have $\mathcal{C}_j^{\text{hard}}(k) \cap \mathcal{C}_j^{\text{soft}}(k) = \emptyset$ and $\mathcal{C}_j^{\text{hard}}(k) \cup \mathcal{C}_j^{\text{soft}}(k) = \mathcal{C}_j(k)$. Let $a_{ij}(k)$ be the duration of operation $i \in \mathcal{C}_j(k)$ plus the inter-process or connection time, if any, between operation i and j.

Now we have the following constraints for the starting time $x_j(k)$ of operation j in the kth cycle:

- **planning constraint**: Operation j cannot start before the planned starting time $d_j(k)$:
$$x_j(k) \geqslant d_j(k) \ .$$
- **hard synchronization**: Operation j can only start if the predecessor operations with hard synchronization conditions are completed:
$$x_j(k) \geqslant x_i(k - 1_{ij}^*(k)) + a_{ij}(k) \quad \text{for each } i \in \mathcal{C}_j^{\text{hard}}(k),$$
with $1_{ij}^*(k)$ defined by: operation j in cycle k has operation i in cycle $k - 1_{ij}^*(k)$ as its predecessor.
- **soft synchronization**: For predecessor operations with soft synchronization conditions we can either wait until they have been completed or break the synchronization. So for each operation $i \in \mathcal{C}_j^{\text{soft}}(k)$ we have
$$x_j(k) \geqslant x_i(k - 1_{ij}^*(k)) + a_{ij}(k) \quad \text{if the synchronization takes place,}$$
$$\text{and } x_j(k) < x_i(k - 1_{ij}^*(k)) + a_{ij}(k) \quad \text{if the synchronization is broken.}$$
If we introduce a control variable $v_{ij}(k) \geqslant 0$, we can combine these equations into one equation:
$$x_j(k) \geqslant x_i(k - 1_{ij}^*(k)) + a_{ij}(k) - v_{ij}(k) \ ,$$
where $v_{ij}(k)$ can be used to guarantee or to break the synchronization.

Since we let operation j start as soon as all synchronization conditions are satisfied, we have

$$\begin{aligned} x_j(k) = \max\big(&d_j(k), \\ &\max_{i \in \mathcal{C}_j^{\text{hard}}(k)} (x_i(k - 1_{ij}^*(k)) + a_{ij}(k)), \\ &\max_{i \in \mathcal{C}_j^{\text{soft}}(k)} (x_i(k - 1_{ij}^*(k)) + a_{ij}(k) - v_{ij}(k))\big) \ . \end{aligned} \quad (4)$$

Remark 1. The relation between (4) and the max-plus-linear model (1)–(2) or (3) becomes clearer if we rewrite (4) using $\oplus$ and $\otimes$. This yields

$$\begin{aligned} x_j(k) = d_j(k) \oplus \bigoplus_{i \in \mathcal{C}_j^{\text{hard}}(k)} a_{ij}(k) \otimes x_i(k - 1_{ij}^*(k)) \oplus \\ \bigoplus_{i \in \mathcal{C}_j^{\text{soft}}(k)} a_{ij}(k) \otimes x_i(k - 1_{ij}^*(k)) \otimes w_{ij}(k) \end{aligned} \quad (5)$$

where $w_{ij}(k) = -v_{ij}(k)$. The first term on the right-hand side of (5) corresponds to the input term $B(k) \otimes u(k)$ with the max-plus-algebraic identity matrix for $B(k)$, the second term corresponds to $A(k) \otimes x(k)$, and the third term can be considered as a "bilinear" extension of $A(k) \otimes x(k)$ with a max-plus-algebraic product of $x(k)$ and the input $w(k)$ (which is a column vector containing the $w_{ij}(k)$'s). So DES with soft and hard constraints can in fact be modeled using a max-plus-*bi*linear model.

Define $t_{ij}^{\text{slack}}(k)$ as the slack time of the completion of operation $i \in \mathcal{C}_j^{\text{soft}}(k)$ (inter-process or connection time included) w.r.t. the actual starting time of operation j in cycle k:

$$t_{ij}^{\text{slack}}(k) = x_i(k - 1_{ij}^*(k)) + a_{ij}(k) - x_j(k) \ .$$

If $t_{ij}^{\text{slack}}(k) \leqslant 0$ then the synchronization is completely guaranteed. If $t_{ij}^{\text{slack}}(k) > 0$ then the synchronization is broken and a penalty has to be paid. If $t_{ij}^{\text{slack}}(k)$ is small, say smaller than some value $t_{ij}^{\max}(k)$, we could assume that a large part of the intermediate products of operation i has already been finished (if the system works in batches) and only a small amount of intermediate products have to be bought from a third party [1] or that the completion of operation i can be performed by a third party at a low cost [2]. Therefore, we define the cost of a broken synchronization as the following piecewise-linear function:

$$J_{\text{broken}}(t_{ij}^{\text{slack}}(k), t_{ij}^{\max}(k), c_{ij}^{\text{broken}}(k)) = \begin{cases} 0 & \text{if } t_{ij}^{\text{slack}}(k) \leqslant 0, \\ \dfrac{c_{ij}^{\text{broken}}(k)}{t_{ij}^{\max}(k)}\, t_{ij}^{\text{slack}}(k) & \text{if } 0 < t_{ij}^{\text{slack}}(k) \leqslant t_{ij}^{\max}(k), \\ c_{ij}^{\text{broken}}(k) & \text{if } t_{ij}^{\text{slack}}(k) > t_{ij}^{\max}(k) \, . \end{cases} \quad (6)$$

3. MPC FOR DES WITH HARD AND SOFT SYNCHRONIZATION

We define the following cost function over a given prediction horizon N_{p} for a given input sequence $\{v_{ij}(k+l)\}_{i;j;l=0,\ldots,N_{\text{p}}-1}$:

$$\begin{aligned} J_{\text{cost}}(k) = \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} |\hat{x}_j(k+l) - d_j(k+l)| + \\ \lambda \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} \sum_{i \in \mathcal{C}_j^{\text{soft}}(k)} J_{\text{broken}}(\hat{t}_{ij}^{\text{slack}}(k+l), \end{aligned}$$

[1] For the sake of simplicity, we assume that the third party always has the parts in storage and that the delivery time is negligible.

[2] Again, we assume for the sake of simplicity that the third party can allocate enough resources to this task so that the execution time is negligible.

$$t_{ij}^{\max}(k+l), c_{ij}^{\text{broken}}(k+l)) \ , \qquad (7)$$

where $\lambda > 0$ is a weighting factor and where $\hat{x}(k+l)$ is the predicted state for the $(k+l)$th operation cycle of the system, and $\hat{t}_{ij}^{\text{slack}}(k+l)$ is the predicted value for the slack t_{ij}^{slack} in the $(k+l)$th cycle. Equation (4) can be used to predict $\hat{x}(k+l)$ and $\hat{t}_{ij}^{\text{slack}}(k+l)$ for the given input sequence. The cost function $J_{\text{cost}}(k)$ has two components: the first tries to keep the operations starting on schedule, whereas the second penalizes broken synchronizations. The factor λ determines the trade-off or relative weight of the two components of the MPC cost function.

Now we consider the following controller design problem, which will be called the *soft synchronization MPC problem* at cycle k:

$$\min_{v_{ij}(k),\ldots,v_{ij}(k+N_{\text{p}}-1)} J_{\text{cost}}(k)$$
subject to (4) and $v_{ij}(k+l) \geqslant 0$ for all i,j and $l = 0,\ldots,N_{\text{p}}-1$.

In addition, to reduce the number of control variables we can — just as in conventional MPC — introduce a control horizon N_{c} ($\leqslant N_{\text{p}}$) and set

$$v_{ij}(k+l) = v_{ij}(k+N_{\text{c}}-1) \quad \text{for } l \geqslant N_{\text{c}}\,. \qquad (8)$$

This condition can be interpreted as follows: if after N_{c} cycles the delays have died out (i.e., it is not necessary to break synchronizations anymore or equivalently, $v_{ij}(k+N_{\text{c}}) = 0$ for all i,j), then we do not break any synchronizations in the subsequent cycles either. On the other hand, if the delays are still such that a synchronization should be broken in cycle $k+N_{\text{c}}$, then we will also break these synchronizations in the subsequent cycles.

Just like in conventional MPC we use a moving horizon approach, i.e., the soft synchronization MPC problem is solved for each operation cycle of the system, then the computed controls for that cycle are applied, and meanwhile the model is updated, and the computation is performed again for the next cycle. This implies that we can also include predictable future delays (due to incidents, broken machines, or maintenance) into our prediction model. The parameter N_{p} should be chosen such that it covers the (expected) period over which the delays will die out. The choice of N_{c} mainly depends on the computational complexity of the problem. For small-sized systems we can take N_{c} rather large, whereas for large-sized systems a small N_{c} will be necessary to be able to compute the MPC solution sufficiently fast (i.e., before the start of the next operation cycle).

In general each step of the soft synchronization MPC problem leads to a non-convex nonlinear optimization problem, which can be solved using, e.g., multi-start sequential quadratic programming. In the next section we will present an alternative approach to compute the optimal MPC control input which is based on a mathematical programming problem called the extended linear complementarity problem (ELCP).

4. LINK WITH THE ELCP

The ELCP is defined as follows (De Schutter and De Moor, 1995):

Given $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{q\times n}$, $c \in \mathbb{R}^p$, $d \in \mathbb{R}^q$ and $\phi_1,\ldots,\phi_m \subseteq \{1,\ldots,p\}$, find $z \in \mathbb{R}^n$ such that

$$\prod_{i\in\phi_j} (Az-c)_i = 0 \quad \text{for } j = 1,\ldots,m, \qquad (9)$$

$$Az \geqslant c, \ \ Bz = d \ . \qquad (10)$$

This can be interpreted as follows: each set ϕ_j corresponds to a group of inequalities of $Az \geqslant c$ and in each group at least one inequality should hold with equality, i.e., for each j there should exist an index $i \in \phi_j$ such that $(Az-c)_i = 0$.

Proposition 2. The evolution equations and the constraints of the soft synchronization MPC problem can be recast as an ELCP.

PROOF. Clearly, the non-negativity constraint on $v_{ij}(k)$ and the control horizon constraint fit the ELCP framework. Now we show that (4) can also be written as an ELCP. This will be done by showing that an expression of the form $x_j(k) = \max(d_j(k), e_j(k), f_j(k))$ is an ELCP. If we then add the conditions that $e_j(k)$ and $f_j(k)$ should be equal to the second term and the third term of the right-hand side of (4) and if we take into account that the merge of two ELCPs is also an ELCP, we have recursively shown that (4) can also be written as an ELCP. The condition $x_j = \max(d_j, e_j, f_j)$ can be rewritten as $x_j - d_j \geqslant 0$, $x_j - e_j \geqslant 0$, $x_j - f_j \geqslant 0$, with $x_j = d_j$ or $x_j = e_j$ or $x_j = f_j$. The latter condition can be rewritten as $(x_j - d_j)(x_j - e_j)(x_j - f_j) = 0$. Hence, we have obtained an ELCP. □

The solution set of an ELCP is the union of a subset of faces of the polyhedron defined by (10). So it is the union of convex sets. The ELCP algorithm of (De Schutter and De Moor, 1995) yields a parametric description of the solution set in which each face is presented by its vertices. The optimal MPC strategy can now be obtained by determining for each face the combination of the parameters for which $J_{\text{cost}}(k)$ reaches a global minimum (this is an optimization over a convex set) and afterwards selecting the overall minimum. The advantage of this approach compared to straightforward nonlinear constrained optimization is that in the ELCP approach we have to solve

Table 1. The nominal processing times and the scheduled starting times.

Buffer/Machine B_j/M_j	Processing time $t_j(k)$	Scheduled starting time $d_j(k)$ (modulo $T = 15$)
B_1	0	0
M_2	10	1
M_3	13	11
M_4	14	11
M_5	4	25

a sequence of optimization problems with a convex feasible set instead of one big problem with a non-convex feasible set. Optimization problems with a convex feasible set (albeit with a non-convex objective function) are easier to solve numerically than problems with a non-convex feasible set.

5. WORKED EXAMPLE

Consider the production system of Figure 2. There is one buffer B_1 and 4 machines M_2, M_3, M_4 and M_5. During each cycle a batch of raw material is sent from B_1 to M_1 and M_3. The batch of intermediate products of M_1 is partially sent to M_2 for further processing and partially sent to M_3 where assembly takes place with the material coming from B_1. The intermediate products of M_2 and M_3 are sent to M_4 for the final assembly. We assume that B_1 has a sufficiently large inventory or is regularly refilled, so that it never starves.

Let $x_1(k)$ be the time instant at which buffer B_1 releases a batch of raw material for the kth time, and let $x_j(k)$ be the time instant at which M_j starts working for the kth time. The nominal processing times $t_j(k)$ and the time schedule are given in Table 1. All the times in this example will be expressed in minutes. The length of one cycle of the production system operation is $T = 15$. So $d_j(k) = d_j(1) + (k-1)T$. All the transportation or interprocess times are assumed to be negligible except for $t_{12}(k) = 1$ and $t_{14}(k) = 4$ for all k. The first operation cycle starts at time $t = 0$. A machine starts working on a new batch as soon as all the required material is available and as soon as the machine is idle (i.e., the machine has finished the previous batch). We assume that there are buffers with a large capacity between the input buffer B_1 and the machines, and between the machines themselves, so that no internal buffer overflow can occur. At the beginning of the first cycle all the machines are empty.

Now we consider two types of synchronization constraints:

- **hard synchronization**: The synchronization constraint on M_4 for the raw material going from B_1 to M_4 is hard, and the same holds for the constraint that a machine can only start working on a new batch if it is idle.
- **soft synchronization**: The synchronization constraint on M_4 for the intermediate products coming from M_2 is soft, and the same holds for the synchronization constraint on M_5 for the intermediate products coming from M_3 and M_4.

We set $t_{24}^{\max}(k) = 8$, $t_{35}^{\max}(k) = t_{45}^{\max}(k) = 10$, and $c_{24}^{\text{broken}}(k) = c_{35}^{\text{broken}}(k) = c_{45}^{\text{broken}}(k) = 10$ for all k.

Now we write down the equations that describe the evolution of the $x_j(k)$'s. Since there are no synchronization constraints at buffer B_1 we have

$$x_1(k) = d_1(k) = 0 + (k-1)T \ .$$

The kth batch of raw material that leaves B_1 at time instant $x_1(k)$ will reach M_2 at time $t = x_1(k) + t_{12}(k) = x_1(k) + 1$. If $k = 0$ then M_2 is idle and can immediately start processing the batch of raw material as soon as it arrives. If $k > 0$ then we have to wait until M_2 has finished processing the previous batch (i.e., the $(k-1)$th batch), which will happen at $t = x_2(k-1) + t_2(k-1)$. Hence, we have [3]

$$x_2(k) = \max(1 + (k-1)T, x_1(k) + 1, x_2(k-1) + t_2(k-1)) \ .$$

Using a similar reasoning we find

$$x_3(k) = \max(11 + (k-1)T, x_2(k) + t_2(k), x_3(k-1) + t_3(k-1)) \ .$$

Machine M_4 can start processing the kth batch as soon as the scheduled starting time $d_4(k) = 11 + (k-1)T$ has passed, the raw material has arrived (which happens at $t = x_1(k) + t_{14}(k) = x_1(k) + 4$), the machine is idle (which happens at $t = x_4(k-1) + t_4(k-1)$), and the intermediate products from machine M_2 has arrived (which happens at $t = x_2(k) + t_2(k)$). However, since the last condition is a soft synchronization condition, we introduce a control variable $v_{24}(k)$ to break the synchronization if necessary. Hence, we have

$$x_4(k) = \max(11 + (k-1)T, x_1(k) + 4, x_4(k-1) + t_4(k-1), x_2(k) + t_2(k) - v_{24}(k)) \ .$$

Analogously, we find

$$\begin{aligned} x_5(k) = \max(&25 + (k-1)T, \\ &x_5(k-1) + t_5(k-1), \\ &x_3(k) + t_3(k) - v_{35}(k), \\ &x_4(k) + t_4(k) - v_{45}(k)) \end{aligned}$$

$$y(k) = x_5(k) + t_5(k)$$

where $y(k)$ is the time instant at which the kth batch of finished products leaves the system.

Let us now assume that all processing times are nominal (cf. Table 1) except for $t_2(1) = 15$. Let $N_c = 4$, $N_p = 6$, and $\lambda = 2$.

---

[3] The equation also holds for $k = 0$ if we set $x_2(0) = -\infty$.

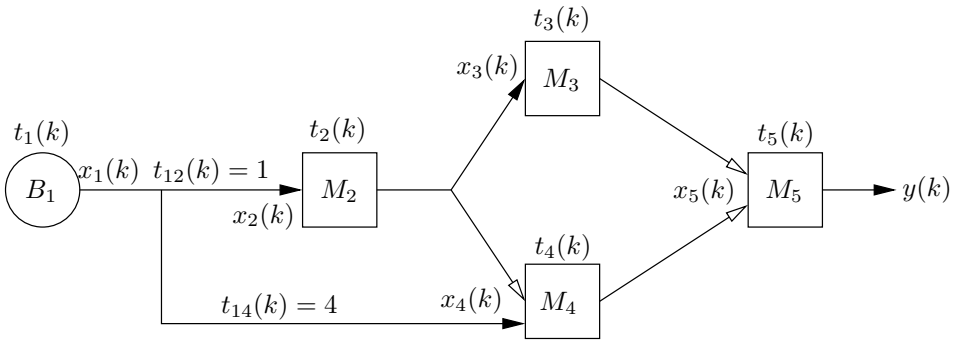

Fig. 2. The production system of the example. The filled arrows represent hard synchronization constraints, and the open arrows represent soft synchronization constraints.

If we do not break any synchronizations then we find maximal delays w.r.t. the time schedule of 5 minutes in the first cycle (for M_3, M_4 and M_5), 4 in the second cycle (for M_4 and M_5), 3 in the third cycle (for M_4 and M_5), 2 in the fourth cycle (for M_4 and M_5), and 1 minute in the fifth cycle (for M_4 and M_5). In the sixth cycle the machines operate again according to schedule. If we do not break any synchronizations, then the value of the total MPC cost function is 39.

If we compute the optimal MPC control input for $k = 1$, we find the following solution: partially (37.5 %) break the synchronization $M_2 \to M_4$ in the first cycle, and do not break any other synchronizations. More specifically, we have $v_{24}(1) = 3$. If we apply this control strategy, then we find a maximal delay w.r.t. the time schedule of 5 minutes in the first cycle, 3 in the second cycle and 1 in the third cycle (all for M_3). In the fourth cycle all operations start again on schedule. The corresponding value of the total MPC cost function is 25.5.

For $\lambda = 0.5$ we get the following solution: partially (62.5 %) break the synchronization $M_2 \to M_4$ in the first cycle, and partially break the synchronization $M_3 \to M_5$ during the first (40 %) and the second cycle (20 %). More specifically, we have $v_{24}(1) = 5$, $v_{35}(1) = 4$ and $v_{35}(2) = 2$. The maximal delays are the same as for $\lambda = 2$ although the total sum of all the delays is less: 9 versus 18.

6. CONCLUSIONS

We have presented an MPC-like control design method for a class of discrete-event systems with both soft and hard synchronization constraints. The control action consists in breaking certain soft synchronization conditions to prevent delays from accumulating, but this can only be done at a certain cost. We have shown that the behavior of such systems can be described by a max-plus-bilinear model. We have also shown that the resulting optimization problem can be solved using ELCPs. Furthermore, due to the use of a moving horizon strategy and a control horizon this method can be used in on-line applications and it can deal with (predicted) changes in the system parameters. So if we can predict the delays that will occur due to an incident, a machine breakdown, or maintenance works, then we can include this information when determining the optimal control input for the next cycles of the operation of the system.

Topics for future research include the development of tuning rules and of efficient algorithms to solve the soft synchronization MPC problem, and the inclusion of modeling errors and/or (bounded) uncertainty in the prediction model.

ACKNOWLEDGEMENTS

Research partially sponsored by the European Community TMR project ALAPEDES and by the FWO Research Community ICCoS.

REFERENCES

Baccelli, F., G. Cohen, G.J. Olsder and J.P. Quadrat (1992). *Synchronization and Linearity*. John Wiley & Sons. New York.

Camacho, E.F. and C. Bordons (1995). *Model Predictive Control in the Process Industry*. Springer. Berlin, Germany.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalized predictive control – Part I. The basic algorithm. *Automatica* **23**(2), 137–148.

Cohen, G., D. Dubois, J.P. Quadrat and M. Viot (1985). A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Trans. on Aut. Contr.* **30**(3), 210–220.

Cuninghame-Green, R.A. (1979). *Minimax Algebra*. Vol. 166 of *Lect. Notes in Econ. and Math. Syst.* Springer. Berlin, Germany.

De Schutter, B. and B. De Moor (1995). The extended linear complementarity problem. *Math. Prog.* **71**(3), 289–325.

De Schutter, B. and T. van den Boom (2000). Model predictive control for max-plus-linear discrete event systems. Accepted for publication in *Automatica*, vol. 37, no. 7, July 2001.

García, C.E., D.M. Prett and M. Morari (1989). Model predictive control: Theory and practice — A survey. *Automatica* **25**(3), 335–348.