Delft University of Technology

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

Technical report bds:01-01

MPC for discrete-event systems with soft and hard synchronisation constraints*

B. De Schutter and T.J.J. van den Boom

If you want to cite this report, please use the following reference instead:
B. De Schutter and T.J.J. van den Boom, "MPC for discrete-event systems with soft and hard synchronisation constraints," *International Journal of Control*, vol. 76, no. 1, pp. 82–94, Jan. 2003. doi:10.1080/0020717021000049188

Control Systems Engineering
Faculty of Information Technology and Systems
Delft University of Technology
Delft, The Netherlands
Current URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/01_01.html

MPC for discrete-event systems with soft and hard synchronisation constraints

B. De Schutter and T.J.J. van den Boom

Control Systems Engineering, Faculty of Information Technology and Systems
Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
tel: +31-15-278.51.13, fax: +31-15-278.66.79
email: {b.deschutter,t.j.j.vandenboom}@its.tudelft.nl

Revised version — June 2002

Abstract

Discrete-event systems with only synchronisation and no concurrency, also known as timed event graphs or (max,+)-linear systems, have been studied by several authors. The synchronisation constraints that arise in these discrete-event systems are hard, i.e., they cannot be broken under any circumstance. In this paper we consider a more extended class of discrete-event systems with both hard and soft synchronisation constraints, i.e., if necessary, some synchronisation conditions may be broken, but then a penalty is incurred. We show how the model predictive control (MPC) framework, which is a very popular controller design method in the process industry, can be extended to this class of discrete-event systems. In general, the MPC control design problem for discrete-event systems with soft and hard synchronisation constraints leads to a nonlinear non-convex optimisation problem. We show that the optimal MPC strategy can also be computed using an extended linear complementarity problem.

Keywords: model predictive control, discrete-event systems, soft synchronisation

1 Introduction

1.1 Model predictive control

Model predictive control (MPC) has shown to respond effectively to control demands imposed by tighter product quality specifications, increasing productivity demands, new environmental regulations, and fast changes in the market. As a result, MPC is now widely accepted in the process industry. There are several other reasons why MPC is probably the most applied advanced control technique in this industry:

- MPC is a model based controller design procedure that can easily handle multi-input multi-output processes.
- It is an easy-to-tune method: in principle only three parameters have to be tuned.
- It can handle constraints on the inputs and the outputs of the process in a systematic way during the design and the implementation of the controller.

- MPC can handle structural changes, such as sensor or actuator failures, and changes in system parameters or system structure, by adapting the model and by using a moving horizon approach, in which the model and the control strategy are regularly updated.

Conventional MPC uses discrete-time models (i.e., models consisting of a system of difference equations). In this paper we extend and adapt the MPC framework to discrete-event systems with soft and hard synchronisation constraints while *retaining the advantages of MPC* listed above. 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. For more information on MPC we refer the interested reader to (Allgöwer *et al.*, 1999; Camacho and Bordons, 1995; Maciejowski, 2002) and the references therein.

1.2 Discrete event systems

Flexible manufacturing systems, telecommunication networks, parallel processing systems, traffic control systems, and logistic systems. This kind of systems are typical examples of discrete-event systems (DESs). One of the most characteristic features of a DES is that its dynamics are event-driven as opposed to time-driven: the behaviour of a DES is governed by events rather than by ticks of a clock. An event corresponds to the start or the end of an activity. For a production system possible events are: the completion of a part on a machine, a machine breakdown, or a buffer becoming empty. In general the dynamics of DESs are characterised by 'synchronisation' and 'concurrency' (Baccelli *et al.*, 1992): Synchronisation requires the availability of several resources at the same time (e.g., before we can assemble a product on a machine, the machine has to be idle and the various parts have to be available). Concurrency appears when at a certain time a user has to choose among several resources (e.g., in a production system a job may be executed on one of the several machines that can handle that job and that are idle at that time). For more information on DES the interested reader is referred to (Baccelli *et al.*, 1992; Cassandras and Lafortune, 1999; Ho, 1992) and the references therein.

In general, the models that describe the behaviour of a DES are nonlinear. However, there is a class of DESs that can be described by a model that is 'linear' in the (max,+) algebra, which has maximisation and addition as its basic operations (Baccelli *et al.*, 1992; Cuninghame-Green, 1979). These '(max,+)-linear' DESs can be characterised as the class of DESs in which only synchronisation and no concurrency occurs; so typical examples are serial production lines, production systems with a fixed routing schedule, and railway networks with rigid connection constraints.

1.3 Overview of the paper

The synchronisation constraints for (max,+)-linear DESs are 'hard': these constraints should always be met. In this paper we consider an extension of the (max,+)-linear DESs: we propose a modelling and control framework for a class of DESs with both *soft* and *hard* synchronisation constraints. So 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

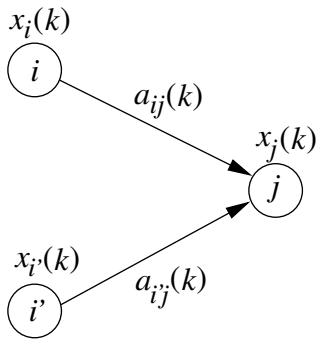

Figure 1: An operation and its predecessors.

synchronisation constraints are hard, then an operation can only start if all the 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 synchronisation constraints so that we can start the operation anyway if further delay would lead to too large deviations from the planned schedule. However, in that case we will incur a penalty cost (e.g., if the intermediary output products of the delayed predecessor operation have to be bought from a third party). Another example is that of a railway operation, where a train should give pre-defined connections to other trains. However, if some of these trains have a too large delay, then it is sometimes better — from a global performance point of view — to let the train depart anyway in order to prevent an accumulation of delays in the network. Of course, missed connections lead to a penalty due to dissatisfied passengers.

In (De Schutter and van den Boom, 2001) we have already extended MPC to the class of (max,+)-linear DESs. In this paper we will further extend the MPC framework to DESs with both hard and soft synchronisation constraints. In general, the proposed MPC approach will lead to a hard non-convex nonlinear optimisation problem. However, we will show that the trajectories of a DES with hard and soft synchronisation constraints can be described by an extended linear complementarity problem (ELCP), for which recently more efficient solution techniques (based on mixed integer programming and on linear complementarity problems) have been developed. We will also discuss an extended modelling framework with controlled changes in the processing times of the operations. Related work is described in (Gokbayrak and Cassandras, 2000; Heidergott and de Vries, 2001).

2 DESs with soft synchronisation constraints

Consider a system with a cyclic operation[1] and in which some operations depend on some predecessor operations (cf. figure 1). 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[2].

[1] This may occur in, e.g., a manufacturing system or a railway operation. In the remainder of the paper, we will mainly focus on manufacturing systems. Note however that the results can also be applied to railway operations.

[2] If there is no scheduled starting time, we can set $d_j(k) = -\infty$ and also omit the corresponding term in the definition of the MPC cost function (3).

The set $\mathcal{C}_j(k)$ of predecessor operations for operation j in the kth cycle can be divided into in a set $\mathcal{C}_j^{\text{hard}}(k)$ of predecessor operations with hard synchronisation conditions (i.e., operation j can only be started if these predecessor operations have been finished, no matter how much time they are delayed), and a set $\mathcal{C}_j^{\text{soft}}(k)$ of predecessor operations with soft synchronisation conditions (i.e., operation j can be started before these predecessor operations have been finished, but then a penalty has to be paid). We have $\mathcal{C}_j^{\text{hard}}(k) \cap \mathcal{C}_j^{\text{soft}}(k) = \emptyset$. Let $c_{ij}^{\text{broken}}(k)$ be the (maximal) cost associated with a broken synchronisation (see (2) for a refinement).

Let $a_{ij}(k)$ be the duration of operation $i \in \mathcal{C}_j(k)$ plus the inter-process time[3] — if any — between operation i and j. The inter-process time could, e.g., be a transportation time needed to transport the intermediate products of operation i to the location where operation j takes place, a set-up time between the operations, etc.

Now we have the following constraints for the starting time of operation j in cycle k:

- **planning constraint**:
  Operation j can only start on or after the planned starting time $d_j(k)$:

$$x_j(k) \geqslant d_j(k) \ .$$

- **hard synchronisation constraints**:
  Operation j can only start if the predecessor operations with hard synchronisation conditions have been completed:

$$x_j(k) \geqslant x_i(k - \delta_{ij}^*(k)) + a_{ij}(k) \quad \text{for each } i \in \mathcal{C}_j^{\text{hard}}(k),$$

  where $\delta_{i,j}^*(k)$ denotes the cycle delay between operations i and j for the kth cycle, i.e., operation j in cycle k has operation i in cycle $k - \delta_{i,j}^*(k)$ as its predecessor (see also the example in Section 5).

- **soft synchronisation constraints**:
  For predecessor operations with soft synchronisation conditions we can either wait until they have been completed or break the synchronisation. So for each $i \in \mathcal{C}_j^{\text{soft}}(k)$ we have

$$\begin{aligned} & x_j(k) \geqslant x_i(k - \delta_{ij}^*(k)) + a_{ij}(k) \quad && \text{if the synchronisation takes place,} \\ \text{or} \quad & x_j(k) < x_i(k - \delta_{ij}^*(k)) + a_{ij}(k) \quad && \text{if the synchronisation is broken.} \end{aligned}$$

  If we introduce a control variable $v_{ij}(k) \geqslant 0$, then we can combine these two equations into one equation:

$$x_j(k) \geqslant x_i(k - \delta_{ij}^*(k)) + a_{ij}(k) - v_{ij}(k) \ ,$$

  where $v_{ij}(k)$ can be used to guarantee or to break the synchronisation.

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

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

[3]For a railway network the inter-process time could include, e.g., the change-over time.

In a nominal, well-defined time schedule the term $d_j(k)$ in (1) will be the largest. However, if due to unforeseen circumstances (an incident, a machine breakdown, a late departure, etc.) operation i has a delay, the corresponding term can become larger than the others.

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 time included) w.r.t. the actual starting time of operation j in the kth cycle:

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

If $t_{ij}^{\text{slack}}(k) \leqslant 0$ then the synchronisation is completely guaranteed. On the other hand, if $t_{ij}^{\text{slack}}(k) > 0$ then the synchronisation is broken and a penalty has to be paid. Note that $t_{ij}^{\text{slack}}(k)$ is a function of the control variable $v_{ij}(k)$ via $x_j(k)$. If $t_{ij}^{\text{slack}}(k)$ is small and if we have a production system that works in batches, then we could assume that a large part of the intermediate products of operation i has already been finished and can be used in operation j. So in that case only a small amount of intermediate products have to be bought from a third party[4]. Alternatively, the completion of operation i could be performed by a third party at a low cost[5]. Therefore, we define the cost of a broken synchronisation as the following piecewise-linear function (see also figure 2) :

$$J_{\text{broken}}(t_{ij}^{\text{slack}}(k), t_{ij}^{\max}(k), c_{ij}^{\text{broken}}(k)) = \begin{cases} 0 & \text{if } t_{ij}^{\text{slack}}(k) \leqslant 0, \\ \dfrac{c_{ij}^{\text{broken}}(k)}{t_{ij}^{\max}(k)}\, t_{ij}^{\text{slack}}(k) & \text{if } 0 < t_{ij}^{\text{slack}}(k) \leqslant t_{ij}^{\max}(k), \\ c_{ij}^{\text{broken}}(k) & \text{if } t_{ij}^{\text{slack}}(k) > t_{ij}^{\max}(k) \end{cases} \quad (2)$$

where $t_{ij}^{\max}(k)$ is the maximal slack time[6] that still allows (partial) use intermediate products of operation i (this time will in general depend on $a_{ij}(k)$).

3 MPC for DESs with hard and soft synchronisation

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

$$J_{\text{cost}}(k) = \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} |\hat{x}_j(k+l) - d_j(k+l)| + \lambda \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} \sum_{i \in \mathcal{C}_j^{\text{soft}}(k+l)} J_{\text{broken}}(\hat{t}_{ij}^{\text{slack}}(k+l), \quad t_{ij}^{\max}(k+l), c_{ij}^{\text{broken}}(k+l)) \ , \quad (3)$$

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 time of

[4]For the sake of simplicity, we assume that the third party always has the parts in storage and that the delivery time is negligible. However, if these assumptions do not hold, the model can be adapted accordingly.

[5]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.

[6]In a railway operations context we could assume that the cost of a broken synchronisation is proportional to the difference between the actual departure time and the scheduled departure time (i.e., fast-running passengers get the connection, but slower ones may lose it). In this case $t_{ij}^{\max}(k)$ corresponds to the delay for which no passengers will succeed in getting on train j, no matter how fast they run.

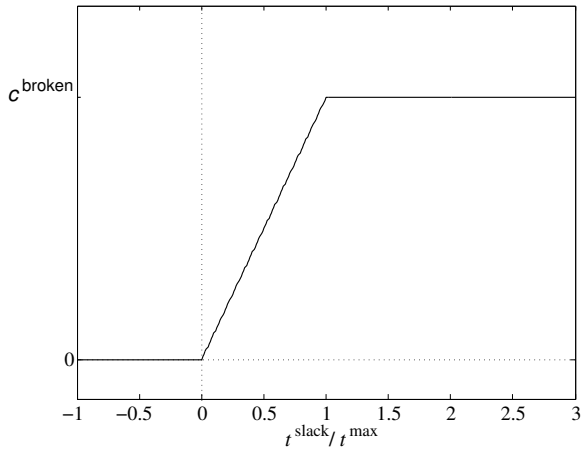

Figure 2: The piecewise-linear cost function J_{broken}.

operation i w.r.t. operation j in the $(k+l)$th cycle. Equation (1) can be used to predict $\hat{x}(k+l)$ and $\hat{t}_{ij}^{\text{slack}}(k+l)$ for the given input sequence. Note that in fact the absolute value in the first term of $J_{\text{cost}}(k)$ may be omitted since $\hat{x}_j(k+l) \geqslant d_j(k+l)$ by (1). The cost function $J_{\text{cost}}(k)$ has two components: the first tries to keep the operations starting on schedule, whereas the second penalises broken synchronisations. 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 synchronisation MPC problem* at cycle k:

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

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

$$v_{ij}(k+l) = v_{ij}(k+N_{\text{c}}-1) \qquad \text{for } l = N_{\text{c}},\ldots,N_{\text{p}}-1\,. \tag{4}$$

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 synchronisations anymore or equivalently, $v_{ij}(k+N_{\text{c}}) = 0$ for all i,j), then we do not break any synchronisations in the subsequent cycles either. On the other hand, if the delays are still such that a synchronisation should be broken in cycle $k+N_{\text{c}}$, then we will also break these synchronisations in the subsequent cycles.

Just like in conventional MPC we use a moving horizon approach, i.e., the soft synchronisation MPC problem is solved for the next 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 subsequent cycle. This implies that we can also include predictable future delays (due to incidents, broken machines, 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 of the system).

In general, each step of the soft synchronisation MPC problem leads to a non-convex nonlinear optimisation problem. This problem can be solved using, e.g., a multi-start local optimisation method such as multi-start sequential quadratic programming. However, this will not always result in a global optimum. In practice, several local optimisation runs with different starting points will be necessary to get a good approximation of the global optimum. Also note that the feasible set of the soft synchronisation MPC problem is non-convex since (1) is a non-convex constraint. In the next section we will present an alternative approach to compute the optimal MPC control input that is based on a mathematical programming problem called the extended linear complementarity problem.

Remark 3.1 If $\hat{t}_{ij}^{\text{slack}}(k+l)$ is nonpositive (or if there is another index i' such that $\hat{t}_{ij}^{\text{slack}}(k+l) > \hat{t}_{i'j}^{\text{slack}}(k+l)$), then $v_{ij}(k+l)$ does not influence the value of the objective function anymore. Therefore, we could extend the MPC cost function with the extra term

$$\eta \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} \sum_{i \in \mathcal{C}_j^{\text{soft}}(k+l)} v_{ij}(k+l) \tag{5}$$

with $\eta > 0$ a small number. In that way, we get the smallest possible values of the $v_{ij}(k+l)$'s. This also enables us to see more clearly which synchronisations are broken or not.

4 Link with the ELCP

The Extended Linear Complementarity Problem (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 m subsets $\phi_1, \ldots, \phi_m$ of $\{1, \ldots, p\}$, find $z \in \mathbb{R}^n$ such that
>
> $$\prod_{i \in \phi_j} (Az - c)_i = 0 \quad \text{for } j = 1, \ldots, m, \tag{6}$$
>
> subject to
>
> $$Az \geqslant c \tag{7}$$
> $$Bz = d \ . \tag{8}$$

Equation (6) represents the complementarity condition of the ELCP, and 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., the corresponding slack variable should be equal to 0. So for each j there should exist an index $i \in \phi_j$ such that $(Az - c)_i = 0$.

The formulation of the ELCP arose from our research on nonlinear resistive networks, DESs ((max,+)-linear systems, (min,max,+) systems, and applications in the (max,+) algebra and the (max,+)-algebraic system theory) and hybrid systems (traffic signal control, and first-order hybrid systems with saturation) (see (De Moor *et al.*, 1992; De Schutter, 2000; De Schutter and van den Boom, 2001; van der Schaft and Schumacher, 2000) and the references therein for more information).

The following proposition gives a link between the soft synchronisation MPC problem and the ELCP:

Proposition 4.1 *The evolution equations and the constraints of the soft synchronisation MPC problem can be recast as an ELCP.*

Proof: Let z be a column vector that contains all $x_{ij}(k)$'s and $v_{ij}(k)$'s. Clearly, the non-negativity constraint on $v_{ij}(k)$ and the control horizon constraint (4) fit the ELCP framework since they correspond to inequalities of the form $Az \geqslant c$ and equalities of the form $Bz = d$ respectively.

Now we show that (1) can also be written as an ELCP. This will be done by showing that a (max,+) expression of the form

$$z_j = \max\big(\max_{i=1,\ldots,\ell_j}(\alpha_{ij} + z_i), \beta_j\big) \tag{9}$$

with $\alpha_{ij}, \beta_j \in \mathbb{R}$ can be recast as an ELCP. Equation (9) can be rewritten as

$$z_j - z_i \geqslant \alpha_{ij} \qquad \text{for } i = 1, \ldots, \ell_j \tag{10}$$

$$z_j \geqslant \beta_j \tag{11}$$

$$z_j = \beta_j \qquad \text{or} \qquad z_j - z_i = \alpha_{ij} \quad \text{for some } i \in \{1, \ldots, \ell_j\} \tag{12}$$

Condition (12) can be rewritten as

$$(z - \beta_j) \cdot \prod_{i=1}^{\ell_j} (z_j - z_i - \alpha_{ij}) = 0 \ ,$$

which is a condition of the form (6). Hence, (9) can be rewritten as an ELCP.

Finally, we note that the merge of several ELCPs is again an ELCP, i.e., N ELCPs of the form

> Given $A_l \in \mathbb{R}^{p_l \times n}$, $B_l \in \mathbb{R}^{q_l \times n}$, $c_l \in \mathbb{R}^{p_l}$, $d_l \in \mathbb{R}^{q_l}$ and m_l subsets $\phi_{l,1}, \ldots, \phi_{l,m_l}$ of $\{1, \ldots, p_l\}$, find $z \in \mathbb{R}^n$ such that
>
> $$\prod_{i \in \phi_{l,j}} (A_l z - c_l)_i = 0 \quad \text{for } j = 1, \ldots, m_l,$$
>
> subject to
>
> $$A_l z \geqslant c_l$$
> $$B_l z = d_l \ ,$$

for $l = 1, \ldots, N$ yield again an ELCP of the form (6)–(8) with

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix},\ B = \begin{bmatrix} B_1 \\ \vdots \\ B_N \end{bmatrix},\ c = \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix},\ d = \begin{bmatrix} d_1 \\ \vdots \\ d_N \end{bmatrix},$$

$p = p_1 + \cdots + p_N$, $q = q_1 + \cdots + q_N$, $m = m_1 + \cdots + m_N$, and $\phi_{s_l^m + j} = \{i + s_l^p \mid i \in \phi_{l,j}\}$ for $l = 1, \ldots, N$ and $j = 1, \ldots, m_l$, where $s_1^m = s_1^p = 0$ and $s_l^m = m_1 + \cdots + m_{l-1}$, $s_l^p = p_1 + \cdots + p_{l-1}$ for $l = 2, \ldots, N$.

As a consequence, the trajectories of a DES with soft and hard synchronisation conditions subject to the MPC constraints can be described by an ELCP. □

Let us now discuss two approaches to compute the optimal soft synchronisation MPC strategy using an ELCP.

- **Approach 1: optimisation over the solution set of the ELCP**:
  In (De Schutter and De Moor, 1995) we have developed an algorithm that yields a parametric description of the solution set of an ELCP. The optimal MPC strategy can now be obtained by determining the combination of the parameters for which the objective function $J_{\text{cost}}(k)$ reaches a global minimum (it can be shown that this corresponds to solving a sequence of optimisation problems with a convex feasible set[7]), and afterwards selecting the overall minimum. The advantage of this approach compared to straightforward nonlinear constrained optimisation of the soft synchronisation MPC problem is that in the ELCP approach we have to solve a sequence of optimisation problems with a convex feasible set instead of one big problem with a non-convex feasible set. Optimisation 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. Note, however, that the algorithm of (De Schutter and De Moor, 1995) to compute the solution set of a general ELCP requires exponential execution times, which implies that the ELCP approach is not feasible if N_{c} is large. Therefore, we will also present another ELCP-based approach, which uses mixed-integer optimisation.

  Our computational experiments have shown that in most cases the determination of the minimum value of the objective functions given above is a well-behaved problem in the sense that using a local minimisation routine (that uses, e.g., sequential quadratic programming) starting from different initial points almost always yields the same numerical result (within a certain tolerance). So for small sized problems or for a small control horizon this ELCP approach is also much faster and yields a better minimum than the straightforward nonlinear optimisation approach.

- **Approach 2: mixed-integer optimisation**:
  In (De Schutter *et al.*, 2002*b*) we have shown that an ELCP with a bounded feasible set[8] can be rewritten as the following mixed-integer problem:

$$\delta \in \{0,1\}^p, \;\; z \in \mathbb{R}^n \tag{13}$$

$$\sum_{i \in \phi_j} \delta_i \leqslant \#\phi_j - 1 \qquad \text{for } j = 1, 2, \ldots, m, \tag{14}$$

$$0 \leqslant (Az - c)_i \leqslant d_i^{\text{upp}} \delta_i \qquad \text{for } i = 1, \ldots, p, \tag{15}$$

$$Bz = d \;\; , \tag{16}$$

  where $\#\phi_j$ denotes the cardinality of the set ϕ_j and where d_i^{upp} is an upper bound for $(Az - c)_i$ over the feasible set of the ELCP[9]. So the trajectories of a DES with soft and

[7] In fact, the objective function $J_{\text{cost}}(k)$ has to be minimised over each face of the solution set of the ELCP.

[8] In general the feasible set of the ELCP that corresponds to (1) is not bounded. However, we already know that the $v_{ij}(k)$'s are bounded from below since they are nonnegative. Moreover, if t_0 is the time instant at which the first cycle of the operation of the system starts and $t_{\text{e},k}$ is a hard upper bound for the end of the kth cycle (e.g., the end time of the daily operation), then we do not change the optimal strategy by adding the constraint $v_{ij}(k) \leqslant t_{\text{e},k} - t_0$. Furthermore, it is easy to verify that the $v_{ij}(k)$'s are bounded, then the starting times $x_j(k)$ are also bounded. Hence, we have an ELCP with a bounded feasible set.

[9] d_i^{upp} can be determined via a linear programming problem: $d_i^{\text{upp}} = \max_{z \in \mathbb{R}^n} \{(Az - c)_i \mid Az \geqslant c, \; Bz = d\}$.

hard synchronisation conditions subject to the MPC constraints can also be described by a mixed-integer problem of the form (13)–(16). The optimal MPC strategy can now be determined by minimising the objective function $J_{\text{cost}}(k)$ subject to (13)–(16) using, e.g., a branch-and-bound or a branch-and-cut method (Cordier *et al.*, 1999; Fletcher and Leyffer, 1998). The main advantage of this approach over (multi-start) local nonlinear optimisation is that we always get the global optimum. A disadvantage might be that the running time of the optimisation might become too large if we have a large number of variables (i.e., a large number of operations n and/or a large control horizon N_{c}). However, this approach is much more efficient and it can deal with much larger problems than Approach 1 (see also (De Schutter *et al.*, 2002*a*)).

Remark 4.2 We can further extend the model of Section 2 and add an extra degree of freedom for the control by also allowing modifications of the process times (but at a cost). We could, e.g., hire additional production units or man power, apply more energy, or speed up the production units (at the expense of faster wear). In this way the processing times $a_{ij}(k)$ can be lowered with respect to their nominal values. This yields an additional control input to prevent the accumulation and propagation of delays, but it also augments the costs.

Let $a_{\text{min},i,j}(k)$ be the minimal duration of operation $i \in \mathcal{C}_j(k)$ when all available extra resources are applied to this operation to their full extent, and let $a_{\text{nom},i,j}(k)$ be the nominal processing time. We introduce an extra control variable $u_{i,j}(k)$ to modify the processing time of operation $i \in \mathcal{C}_j(k)$ in cycle k. We get a model for the extended system by replacing all occurrences of $a_{i,j}(k)$ in (1) by $a_{\text{nom},i,j}(k) - u_{i,j}(k)$ and by adding the extra condition

$$0 \leqslant u_{i,j}(k) \leqslant a_{\text{nom},i,j}(k) - a_{\text{min},i,j}(k) \ . \tag{17}$$

To express the extra costs related to increasing the production rate, we add a term of the form

$$\xi \sum_{l=0}^{N_{\text{p}}-1} \sum_{j=1}^{n} \sum_{i \in \mathcal{C}_j(k+l)} c_{\text{spu},i,j}(k+l) \cdot u_{i,j}(k+l) \tag{18}$$

to the cost function $J_{\text{cost}}(k)$, where ξ is a nonnegative weight parameter that represents the relative importance of the speed-up term with respect to the other terms of the (extended) objective function, and where $c_{\text{spu},i,j}(k)$ characterises the extra cost per unit processing time decrease for operation $i \in \mathcal{C}_j(k)$. This results in an extended model for DESs with soft and hard synchronisation constraints, and in an extended soft synchronisation MPC problem. It is easy to verify that this problem can also be solved using the ELCP.

5 Worked example

In this section we present a worked example that involves a production system with soft and hard synchronisation constraints. First, we describe the set-up and the operation of the system. Next, we illustrate the modelling approach of Section 2 by deriving a model of the system. Finally, we apply MPC on this system in order to illustrate the advantages and extra degrees of freedom in the MPC control offered by considering soft synchronisation constraints.

Consider the production system of figure 3. There is one buffer B_1 and 4 machines M_2, M_3, M_4 and M_5. At the beginning of each production cycle a batch of raw material is sent

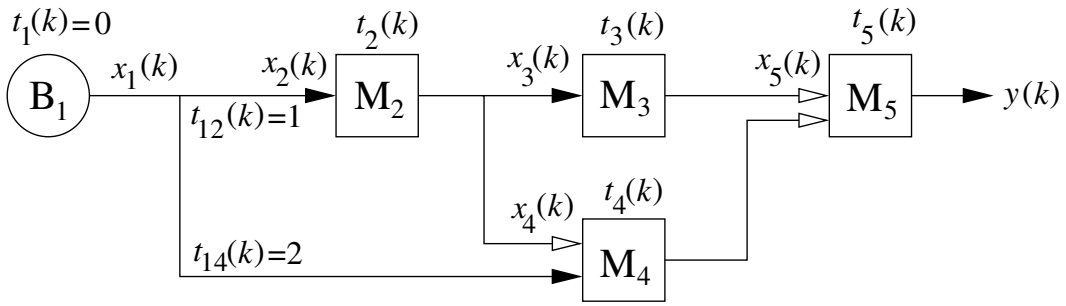

Figure 3: The production system of the example of Section 5. The filled arrows represent hard synchronisation constraints, and the open arrows represent soft synchronisation constraints.

Buffer/Machine B_j/M_j	Processing time $t_j(k)$	Scheduled starting time $d_j(k)$ (modulo $T = 20$)
B_1	0	0
M_2	9	1
M_3	16	10
M_4	18	10
M_5	5	28

Table 1: The nominal processing times and the scheduled starting times for the production system of the example of Section 5.

from the buffer B_1 to machines M_1 and M_3. The batch of intermediate products of M_1 is partially sent to machine M_2 for further processing and partially to machine M_3 where assembly takes place with the material coming from buffer B_1. The intermediate products of M_2 and M_3 are sent to M_4 for the final assembly. We assume that the buffer 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 machine M_j starts working for the kth time for $j = 2, 3, 4, 5$. The nominal processing times $t_j(k)$ and the time schedule of the production system 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 $T = 20$. So the kth scheduled starting time for operation j is given by $d_j(k) = d_j(1) + (k-1)T$. All the transportation or interprocess times are assumed to be negligible except for $t_{12}(k) = 1$ and $t_{14}(k) = 2$ for all k. The first operation cycle of the system starts at time $t = 0$.

Each machine starts processing a new batch as soon the machine is idle (i.e., the machine has finished the previous batch) and as soon as all the required material is available — unless there is a soft synchronisation constraint. Moreover, we assume that there are buffers with a large capacity between the input buffer B_1 and the machines M_2 and M_3, and between M_2 and M_3, M_2 and M_4, M_3 and M_5, and M_4 and M_5, so that no internal buffer overflow can occur. At the beginning of the first cycle all the machines are empty.

There are two types of synchronisations in the production system:

- **hard synchronisations**: The synchronisation constraints on machines M_2 and M_4 for

the raw material coming from the buffer B_1 are hard constraints, and the same holds for the synchronisation constraint on machine M_3 for the material coming from M_2, and for the constraint that a machine can only start working on a new batch if it is idle.

- **soft synchronisations**: The synchronisation constraint on machine M_4 for the intermediate products coming from M_2 is a soft constraint, and the same holds for the synchronisation constraint on machine M_5 for the intermediate products coming from M_3 and M_4.

We set $t_{24}^{\max}(k) = 5$, $t_{35}^{\max}(k) = 10$, $t_{45}^{\max}(k) = 15$, $c_{24}^{\text{broken}}(k) = 10$, $c_{35}^{\text{broken}}(k) = 15$ and $c_{45}^{\text{broken}}(k) = 20$ for all k.

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

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

for all k. The kth batch of raw material that leaves buffer B_1 at time instant $x_1(k)$ will reach machine M_2 at time $t = x_1(k) + t_{12}(k) = x_1(k) + 1$. If $k = 0$ then machine 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 machine M_2 has finished processing the previous batch, which will happen at time $t = x_2(k-1) + t_2(k-1)$. Hence, we have

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

for all k. This equation also holds for $k = 0$ if we set[10] $x_2(0) = -\infty$. Note that — referring to (1) — we have $\delta_{12}^*(k) = 0$ and $\delta_{22}^*(k) = 1$ for all k. Using a similar reasoning we find

$$x_3(k) = \max\big(10 + (k-1)T, x_2(k) + t_2(k), x_3(k-1) + t_3(k-1)\big)$$

for all k with $x_3(0) = -\infty$.

Now consider machine M_4. This machine can start processing the kth batch as soon as the scheduled starting time $d_4(k) = 10 + (k-1)T$ has passed, the raw material has arrived (which happens at time $t = x_1(k) + t_{14}(k) = x_1(k) + 2$), the machine is idle (which happens at time $t = x_4(k-1) + t_4(k-1)$), and the intermediate products from machine M_2 have arrived (which happens at time $t = x_2(k) + t_2(k)$). However, since the last condition is a soft synchronisation condition, we introduce a control variable $v_{24}(k)$ to break the synchronisation if necessary. Hence, we have

$$x_4(k) = \max\big(10 + (k-1)T, x_1(k) + 2, x_4(k-1) + t_4(k-1), x_2(k) + t_2(k) - v_{24}(k)\big)$$

for all k with $x_4(0) = -\infty$. Analogously, we find

$$\begin{aligned} x_5(k) = \max\big(&28 + (k-1)T, x_5(k-1) + t_5(k-1), x_3(k) + t_3(k) - v_{35}(k), \\ &x_4(k) + t_4(k) - v_{45}(k)\big) \\ y(k) = {} & x_5(k) + t_5(k) \end{aligned}$$

for all k with $x_5(0) = -\infty$ and where $y(k)$ is the time instant at which the kth batch of finished products leaves the system.

[10] In fact it is sufficient to select the value of $x_2(0)$ such that $x_2(0) + t_2(0)$ is smaller than $d_2(1) = 1$ or $x_1(1) + 1$. The choice $x_2(0) = -\infty$ guarantees that this condition will always hold.

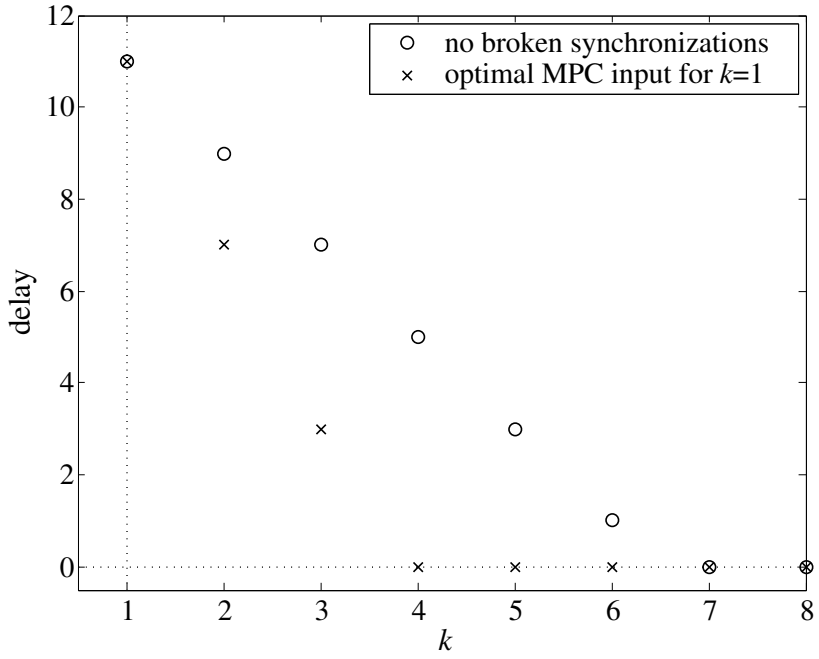

Figure 4: Delays w.r.t. the time schedule of the example of Section 5 for a situation with no broken synchronisations and for the optimal MPC input for $k = 1$ and $N_c = 4$ applied during the period [1,8].

Let us now assume that all processing times are nominal (cf. table 1) except for $t_2(1) = 20$. Let $N_c = 4$, $N_p = 6$, $\lambda = 0.25$ and $\eta = 0.01$.

If we do not break any synchronisations, then we find maximal delays w.r.t. the time schedule of 11 minutes in the first cycle (for machines M_3, M_4 and M_5), and respectively 9, 7, 5, 3, and 1 minutes in the subsequent cycles (for M_4 and M_5). From the seventh cycle on all machines operations start again according to the schedule (see also figure 4). If we do not break any synchronisations, then the value of the total MPC cost function (the term (5) included) is 93.

Let us now compute the optimal MPC control input for one step of the soft synchronisation MPC procedure (i.e., for $k = 1$). Then we find with both the multi-start local nonlinear optimisation approach[11] and the ELCP approaches the following solution: completely break the synchronisation $M_2 \to M_4$ in the first cycle, and partially break the synchronisation $M_3 \to M_5$ during the first, the second and the third cycle. More specifically, we have $v_{24}(1) = 11$, $v_{35}(1) = 9$, $v_{35}(2) = 5$, and $v_{35}(3) = 1$. If we apply this control strategy, then we find a delay w.r.t. the time schedule of 11 minutes in the first cycle, 7 in the second cycle, and 3 in the third cycle (all for machine M_3 — the other machines all operate according to the schedule). In the fourth cycle all operations start again on schedule (see also figure 4). The corresponding value of the total MPC cost function (the term (5) included) is 29.385.

For $\lambda = 0$ (which implies that breaking synchronisations is not penalised at all) we find

[11] We have selected the best result of a sequential quadratic programming algorithm over 20 runs with random starting points. The minimal value of the objective function was 29.38, with mean 29.79 and standard deviation 0.31.

the same result as for $\lambda = 0.25$.
For $\lambda = 2$ (which implies that breaking synchronisations becomes more 'expensive') we find the following MPC solution (for $k = 1$): completely break the synchronisation $M_2 \rightarrow M_4$ in the first cycle, and do not break any other synchronisation. More specifically, we have $v_{24}(1) = 11$. If we apply this control strategy, then we find the same delays w.r.t. the time schedule as for $\lambda = 0.25$ as far as machine M_3 is concerned, and a delay of 9 minutes in the first cycle, 5 in the second cycle, and 1 in the third cycle for machine M_5. The other machines all operate according to the schedule.
If we make breaking synchronisations even more expensive by taking $\lambda = 10$, we get an optimal solution in which no synchronisations should be broken (Note that in general the strategy with no synchronisations broken corresponds to $\lambda \rightarrow \infty$.).

For $N_c = 5$ and $N_c = 6$ (and again $\lambda = 0.25$) we get the same result as for $N_c = 4$, which implies that if we would use the receding horizon approach and apply the first sample of the optimal MPC control input for $k = 1, 2, \ldots, N_p$ during each cycle, we would get the same input sequence as the one given above (provided that the system keeps operating according to the schedule of table 1 and provided that all processing times (except for $t_2(1) = 20$) stay nominal).
For $N_c = 3$ (and $\lambda = 0.25$) we find the same optimal MPC input as for $N_c = 4$, but due to the control horizon constraint (4) we now have $v_{35}(4) = v_{35}(5) = v_{35}(6) = 1 = v_{35}(3)$ instead of $v_{35}(4) = v_{35}(5) = v_{35}(6) = 0$. The corresponding value of the objective function is 29.415 (the difference with the optimal objective value for $N_c = 4$ is entirely due to the term (5), which aims at minimising the $v_{ij}(k)$'s).

6 Conclusions

We have further extended the MPC control design method, which is very popular in the process industry where it is usually based on linear or nonlinear discrete-time models, to a class of DESs with both soft and hard synchronisation constraints. The control action consists in breaking certain soft synchronisation conditions to prevent delays from accumulating, but this can only be done at a certain cost. 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.

We have shown that the optimisation problem that has to be solved in each step of the soft synchronisation MPC problem can be solved using ELCPs (either via optimisation over the parameterised solution set of the ELCP or via mixed-integer optimisation).

Topics for future research are: development of efficient specialised algorithms to solve the soft synchronisation MPC problem, stability issues, determination of tuning rules for the tuning parameters λ, N_p and N_c, and the inclusion of modelling errors and/or (bounded) uncertainty in the prediction model.

Acknowledgement
Research partially funded by the Dutch Technology Foundation STW project 'Model predictive control for hybrid systems' (DMR.5675) and by the European IST project 'Modelling, Simulation and Control of Nonsmooth Dynamical Systems (SICONOS)' (IST-2001-37172).

References

Allgöwer, F., T.A. Badgwell, J.S. Qin, J.B. Rawlings and S.J. Wright (1999). Nonlinear predictive control and moving horizon estimation – An introductory overview. In: *Advances in Control: Highlights of ECC '99* (P.M. Frank, Ed.). London, UK: Springer. pp. 391–449.

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-Verlag. Berlin, Germany.

Cassandras, C.G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publishers. Boston.

Cordier, C., H. Marchand, R. Laundy and L.A. Wolsey (1999). *bc-opt*: A branch-and-cut code for mixed integer programs. *Mathematical Programming, Series A* **86**(2), 335–353.

Cuninghame-Green, R.A. (1979). *Minimax Algebra*. Vol. 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag. Berlin, Germany.

De Moor, B., L. Vandenberghe and J. Vandewalle (1992). The generalized linear complementarity problem and an algorithm to find all its solutions. *Mathematical Programming* **57**, 415–426.

De Schutter, B. (2000). Optimal control of a class of linear hybrid systems with saturation. *SIAM Journal on Control and Optimization* **39**(3), 835–851.

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

De Schutter, B. and T. van den Boom (2001). Model predictive control for max-plus-linear discrete event systems. *Automatica* **37**(7), 1049–1056.

De Schutter, B., W.P.M.H. Heemels and A. Bemporad (2002*a*). Max-plus-algebraic problems and the extended linear complementarity problem — Algorithmic aspects. In: *Proceedings of the 15th IFAC World Congress*. Barcelona, Spain. Paper 728 / T-We-M02.

De Schutter, B., W.P.M.H. Heemels and A. Bemporad (2002*b*). On the equivalence of linear complementarity problems. *Operations Research Letters* **30**(4), 211–222.

Fletcher, R. and S. Leyffer (1998). Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization* **8**(2), 604–616.

Gokbayrak, K. and C.G. Cassandras (2000). Hybrid controllers for hierarchically decomposed systems. In: *Hybrid Systems: Computation and Control* (Proceedings of the 3rdd International Workshop on Hybrid Systems: Computation and Control (HSCC 2000), Pittsburgh, Pennsylvania, March 2000) (N. Lynch and B.H. Krogh, Eds.). Lecture Notes in Computer Science. Berlin, Germany: Springer. pp. 117–129.

Heidergott, B. and R. de Vries (2001). Towards a (max,+) control theory for public transportation networks. *Discrete Event Dynamic Systems: Theory and Applications* **11**(4), 371–398.

Ho, Y.C., Ed.) (1992). *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*. IEEE Press. Piscataway, New Jersey.

Maciejowski, J.M. (2002). *Predictive Control with Constraints*. Prentice Hall. Harlow, England.

van der Schaft, A.J. and J.M. Schumacher (2000). *An Introduction to Hybrid Dynamical Systems*. Vol. 251 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag. London.

A Appendix

A.1 Link with $(\max,+)$-linear models

Recall that DESs with hard synchronisation constraints only can be described by a $(\max,+)$-linear model (Baccelli *et al.*, 1992). In this appendix we show that the model of Section 2 for DESs with soft and hard synchronisation constraints also leads to a (max,+) model (albeit not (max,+)-linear but a special type of (max,+)-bilinear model).

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

$$x_j(k) = \max\Big(\max_{i=1,\ldots,n} \big(\alpha_{ij}(k) + x_i(k-1)\big), \max_{i=1,\ldots,n_u} \big(\beta_{ij}(k) + u_i(k)\big)\Big) \quad \text{for } j = 1,\ldots,n, \tag{A.1}$$

$$y_j(k) = \max_{i=1,\ldots,n} \big(\gamma_{ij}(k) + x_i(k)\big) \quad \text{for } j = 1,\ldots,n_y, \tag{A.2}$$

where n_u and n_y are respectively the number of inputs and outputs. If we use the conventional (max,+)-algebraic symbols $\oplus$ and $\otimes$ for respectively maximisation and addition, (A.1)–(A.2) can be rewritten as[12]

$$x_j(k) = \bigoplus_{i=1}^{n} \alpha_{ij}(k) \otimes x_i(k-1) \oplus \bigoplus_{i=1}^{n_u} \beta_{ij}(k) \otimes u_i(k) \qquad \text{for } j = 1,\ldots,n, \tag{A.3}$$

$$y_j(k) = \bigoplus_{i=1}^{n} \gamma_{ij}(k) \otimes x_i(k) \qquad \text{for } j = 1,\ldots,n_y. \tag{A.4}$$

Note that (A.3) and (A.4) are linear expressions in the operators $\oplus$ and $\otimes$. Therefore, we say that (A.3) and (A.4) are $(\oplus,\otimes)$-linear — or equivalently (max,+)-linear — expressions of the state and the input. As a consequence, the model (A.3)–(A.4) is called (max,+)-linear. The relation between (1) and the (max,+)-linear model (A.3)–(A.4) becomes clearer if we rewrite (1) using the (max,+)-algebraic symbols $\oplus$ and $\otimes$. This yields the following expression:

$$x_j(k) = d_j(k) \oplus \bigoplus_{i \in \mathcal{C}_j^{\text{hard}}(k)} a_{ij}(k) \otimes x_i(k - \delta_{ij}^*(k)) \oplus \bigoplus_{i \in \mathcal{C}_j^{\text{soft}}(k)} a_{ij}(k) \otimes x_i(k - \delta_{ij}^*(k)) \otimes w_{ij}(k) \tag{A.5}$$

where $w_{ij}(k) = -v_{ij}(k)$. The first 'term' (in the (max,+)-algebraic sense) on the right-hand side of (A.5) corresponds to the input term of (A.3) with $\beta_{jj}(k) = 0$ and $\beta_{ij}(k) = -\infty$ for all $i \neq j$ and with $u(k) = d(k)$. The second term of the right-hand side of (A.5) corresponds to the state term of (A.3). The third term of (A.5) can be considered as a 'bilinear' extension of the state term with a (max,+)-algebraic product of the state $x(k)$ and an additional input $w(k)$ (which is a column vector containing the $w_{ij}(k)$'s). So the class of DESs with soft and hard constraints considered in this paper can in fact be modelled using a special type of (max,+)-*bi*linear model.

[12] We use $\bigoplus_{i=1}^{n} a_i$ as a short-hand notation for $a_1 \oplus \cdots \oplus a_n$.

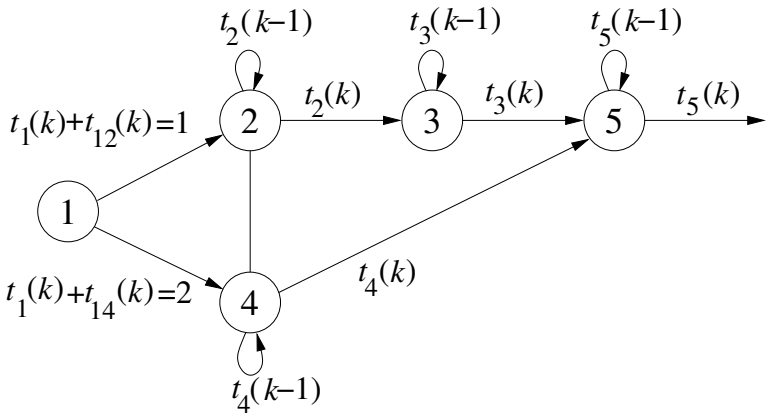

Figure 5: The predecessor relations between the operations of the production systems of Section 5. Operation j corresponds to buffer B_j or machine M_j. The numbers above or next to the arrows denote the duration of the operation (interprocess time included). The cycle delay is 0 everywhere, except for the self-loops, which have a cycle delay of 1.

Remark A.1 We can also combine the soft synchronisation MPC framework of Section 3 and the MPC framework for (max,+)-linear systems of the form (A.1)–(A.2) (see (De Schutter and van den Boom, 2001)) by dropping the time schedule and adding an external input $u(k)$ and an output $y(k)$. If the due dates r for the finished products are known, we could then use the following MPC cost function:

$$J^*_{\text{cost}}(k) = \sum_{l=0}^{N_\text{p}-1} \sum_{j=1}^{n_y} \max(y_j(k+l) - r_j(k+l), 0) - \mu \sum_{l=0}^{N_\text{p}-1} \sum_{j=1}^{n_u} u_j(k+l) + \lambda \sum_{l=0}^{N_\text{p}-1} \sum_{j=1}^{n} \sum_{i \in \mathcal{C}_j^{\text{soft}}(k+l)} J_{\text{broken}}(t_{ij}^{\text{slack}}(k+l), t_{ij}^{\max}(k+l), c_{ij}^{\text{broken}}(k+l))$$

with $\mu \geqslant 0$, and where n_u and n_y are the number of inputs and outputs respectively. The first term of $J^*_{\text{cost}}(k)$ is called the tardiness and penalises late deliveries, whereas the second term aims at maximising the input time instants (which is needed for internal stability). We refer the interested reader to (De Schutter and van den Boom, 2001) for more information and for other possible choices for the first and second term of the objective function.

The resulting modified MPC problem also leads to nonlinear non-convex optimisation problem. Moreover, Proposition 4.1 also holds for this extended soft synchronisation MPC problem.

A.2 The worked example revisited

In this section we have a closer look at the MPC optimisation problem that has to be solved in each operation cycle of the production system of the work example of Section 5.

If we associate each buffer or machine of the production system of figure 1 with an operation, then the graph of figure 5 represents the predecessor/successor relations between the operations. In addition, the sets $\mathcal{C}_j^{\text{hard}}(k)$ and $\mathcal{C}_j^{\text{soft}}(k)$ are listed in table 2.

Operation (B_j/M_j)	$\mathcal{C}_j^{\text{hard}}(k)$	$\mathcal{C}_j^{\text{soft}}(k)$
1	$\emptyset$	$\emptyset$
2	$\{1,2\}$	$\emptyset$
3	$\{2,3\}$	$\emptyset$
4	$\{1,4\}$	$\{2\}$
5	$\{5\}$	$\{3,4\}$

Table 2: The sets that correspond to the hard and soft synchronisation constraints for each operation j. Note that for the synchronisation relations of the form $i \to i$ we have a cycle delay of 1, whereas the other cycle delays are 0.

If we plug the model derived in Section 5 into the formulation of the soft synchronisation MPC problem given in Section 3, then we obtain the following optimisation problem for operation cycle k:

$$\begin{aligned}
\min_{\substack{v_{24}(k),v_{35}(k),v_{45}(k),\\ v_{24}(k+1),v_{35}(k+1),v_{45}(k+1),\ldots,\\ v_{24}(k+N_\text{p}-1),v_{35}(k+N_\text{p}-1),v_{45}(k+N_\text{p}-1)}} \Bigg[& \sum_{l=0}^{N_\text{p}-1} \sum_{j=2}^{5} |\hat{x}_j(k+l) - d_j(k+l)| \\
& + \lambda \sum_{l=0}^{N_\text{p}-1} \Big(J_\text{broken}(\hat{t}_{24}^\text{slack}(k+l), t_{24}^\text{max}(k+l), c_{24}^\text{broken}(k+l)) \\
& \qquad + J_\text{broken}(\hat{t}_{35}^\text{slack}(k+l), t_{35}^\text{max}(k+l), c_{35}^\text{broken}(k+l)) \\
& \qquad + J_\text{broken}(\hat{t}_{45}^\text{slack}(k+l), t_{45}^\text{max}(k+l), c_{45}^\text{broken}(k+l)) \Big) \\
& + \eta \sum_{l=0}^{N_\text{p}-1} \big(v_{24}(k+l) + v_{35}(k+l) + v_{45}(k+l) \big) \Bigg]
\end{aligned}$$

subject to

$$\hat{t}_{24}^\text{slack}(k+l) = \hat{x}_2(k+l) + \hat{t}_2(k+l) - \hat{x}_4(k+l) \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.6}$$

$$\hat{t}_{35}^\text{slack}(k+l) = \hat{x}_3(k+l) + \hat{t}_3(k+l) - \hat{x}_5(k+l) \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.7}$$

$$\hat{t}_{45}^\text{slack}(k+l) = \hat{x}_4(k+l) + \hat{t}_4(k+l) - \hat{x}_5(k+l) \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.8}$$

$$\begin{aligned}
\hat{x}_2(k+l) = \max \big(& 1 + (k+l-1)T, d_1(k+l) + 1, \\
& \hat{x}_2(k+l-1) + \hat{t}_2(k+l-1) \big)
\end{aligned} \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.9}$$

$$\begin{aligned}
\hat{x}_3(k+l) = \max \big(& 10 + (k+l-1)T, \\
& \hat{x}_2(k+l) + \hat{t}_2(k+l), \\
& \hat{x}_3(k+l-1) + \hat{t}_3(k+l-1) \big)
\end{aligned} \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.10}$$

$$\begin{aligned}
\hat{x}_4(k+l) = \max \big(& 10 + (k+l-1)T, \hat{x}_1(k+l) + 2, \\
& \hat{x}_4(k+l-1) + \hat{t}_4(k+l-1), \\
& \hat{x}_2(k+l) + \hat{t}_2(k+l) - v_{24}(k+l) \big)
\end{aligned} \qquad \text{for } l = 0,\ldots,N_\text{p}-1 \tag{A.11}$$

$$\hat{x}_5(k+l) = \max \big(28 + (k+l-1)T,$$

$$\hat{x}_5(k+l-1)+\hat{t}_5(k+l-1),$$
$$\hat{x}_3(k+l)+\hat{t}_3(k+l)-v_{35}(k+l),$$
$$\hat{x}_4(k+l)+\hat{t}_4(k+l)-v_{45}(k+l)\big) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.12)}$$
$$v_{24}(k+l)\geqslant 0 \qquad \text{for } l=0,\dots,N_{\mathrm{c}}-1 \qquad \text{(A.13)}$$
$$v_{35}(k+l)\geqslant 0 \qquad \text{for } l=0,\dots,N_{\mathrm{c}}-1 \qquad \text{(A.14)}$$
$$v_{45}(k+l)\geqslant 0 \qquad \text{for } l=0,\dots,N_{\mathrm{c}}-1 \qquad \text{(A.15)}$$
$$v_{24}(k+l)=v_{24}(k+N_{\mathrm{c}}-1) \qquad \text{for } l=N_{\mathrm{c}},\dots,N_{\mathrm{p}}-1 \qquad \text{(A.16)}$$
$$v_{35}(k+l)=v_{35}(k+N_{\mathrm{c}}-1) \qquad \text{for } l=N_{\mathrm{c}},\dots,N_{\mathrm{p}}-1 \qquad \text{(A.17)}$$
$$v_{45}(k+l)=v_{45}(k+N_{\mathrm{c}}-1) \qquad \text{for } l=N_{\mathrm{c}},\dots,N_{\mathrm{p}}-1, \qquad \text{(A.18)}$$

with $\hat{x}_i(k-1)=x_i(k-1)$ and $\hat{t}_i(k-1)=t_i(k-1)$ for all i, and where $\hat{t}_i(k)$ is an estimate of the duration of operation i. Equations (A.6)–(A.8) define the estimates of the slack times in cycle $k+l$, (A.9)–(A.12) define the estimates of the starting times of the operations[13] in cycle $k+l$, (A.13)–(A.15) represent the non-negativity constraints on the control variables[14], and (A.16)–(A.18) represent the control horizon constraint.

If we want to solve the MPC optimisation problem given above via an ELCP, we first have to transform (A.6)–(A.18) into an ELCP. Since (A.6)–(A.8) and (A.13)–(A.18) are already linear equations — and, hence, fit the ELCP framework —, we will now concentrate on (A.9)–(A.12). Using the reasoning given in the proof of Proposition 4.1 these equations can be rewritten as the following ELCP:

$$\hat{x}_2(k+l)\geqslant 1+(k+l-1)T \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.19)}$$
$$\hat{x}_2(k+l)\geqslant d_1(k+l)+1 \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.20)}$$
$$\hat{x}_2(k+l)\geqslant \hat{x}_2(k+l-1)+\hat{t}_2(k+l-1) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.21)}$$
$$\hat{x}_3(k+l)\geqslant 10+(k+l-1)T \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.22)}$$
$$\hat{x}_3(k+l)\geqslant \hat{x}_2(k+l)+\hat{t}_2(k+l) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.23)}$$
$$\hat{x}_3(k+l)\geqslant \hat{x}_3(k+l-1)+\hat{t}_3(k+l-1) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.24)}$$
$$\hat{x}_4(k+l)\geqslant 10+(k+l-1)T \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.25)}$$
$$\hat{x}_4(k+l)\geqslant \hat{x}_1(k+l)+2 \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.26)}$$
$$\hat{x}_4(k+l)\geqslant \hat{x}_4(k+l-1)+\hat{t}_4(k+l-1) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.27)}$$
$$\hat{x}_4(k+l)\geqslant \hat{x}_2(k+l)+\hat{t}_2(k+l)-v_{24}(k+l) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.28)}$$
$$\hat{x}_5(k+l)\geqslant 28+(k+l-1)T \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.29)}$$
$$\hat{x}_5(k+l)\geqslant \hat{x}_5(k+l-1)+\hat{t}_5(k+l-1) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.30)}$$
$$\hat{x}_5(k+l)\geqslant \hat{x}_3(k+l)+\hat{t}_3(k+l)-v_{35}(k+l) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.31)}$$
$$\hat{x}_5(k+l)\geqslant \hat{x}_4(k+l)+\hat{t}_4(k+l)-v_{45}(k+l) \qquad \text{for } l=0,\dots,N_{\mathrm{p}}-1 \qquad \text{(A.32)}$$

[13] Note that $x_1(k)$ could be eliminated since there are no synchronisation constraints at buffer B_1 and $x_1(k)=d_1(k)$ for all k.

[14] We have already used (A.16)–(A.18) to change the upper bound for the counter l into $N_{\mathrm{c}}-1$ instead of $N_{\mathrm{p}}-1$.

$$\sum_{\ell=0}^{N_\mathrm{p}-1} \Big[\mathrm{sl}_\ell(A.19) \cdot \mathrm{sl}_\ell(A.20) \cdot \mathrm{sl}_\ell(A.21) \;+\; \mathrm{sl}_\ell(A.22) \cdot \mathrm{sl}_\ell(A.23) \cdot \mathrm{sl}_\ell(A.24) \\ +\; \mathrm{sl}_\ell(A.25) \cdot \mathrm{sl}_\ell(A.26) \cdot \mathrm{sl}_\ell(A.27) \cdot \mathrm{sl}_\ell(A.28) \\ +\; \mathrm{sl}_\ell(A.29) \cdot \mathrm{sl}_\ell(A.30) \cdot \mathrm{sl}_\ell(A.31) \cdot \mathrm{sl}_\ell(A.32) \Big], \tag{A.33}$$

where $\mathrm{sl}_\ell(\mathrm{A}.i)$ denotes the difference between the right-hand side and the left-hand side of the linear inequality of $(\mathrm{A}.i)$ for $l = \ell$. Since these differences are always nonnegative due to the inequality constraints, the condition (A.33) implies that in each group of equations (A.19)–(A.21), (A.22)–(A.24), (A.25)–(A.28), (A.29)–(A.32), and for each $\ell \in \{0, 1, \ldots, N_\mathrm{p} - 1\}$ at least one inequality holds with equality, which implies that the maximum in (A.9), (A.10), (A.11), and (A.12) is reached for each $\ell \in \{0, 1, \ldots, N_\mathrm{p} - 1\}$. If we add equations (A.6)– (A.8) and (A.13)–(A.18) to the ELCP (A.19)–(A.33), we get again an ELCP, which describes the (estimated) trajectories of the system up to the prediction horizon, and which can be used to solve the soft synchronisation optimisation problem for cycle k using one of the approaches presented in Section 4.