**Delft University of Technology**

**Delft Center for Systems and Control**

Technical report 05-022

# Modelling and control of discrete event systems using switching max-plus-linear systems*

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

*If you want to cite this report, please use the following reference instead:*
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*, vol. 14, no. 10, pp. 1199–1211, Oct. 2006. doi:10.1016/j.conengprac.2006.02.006

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.deschutter.info/abs/05_022.html

# Modelling and control of discrete event systems using switching max-plus-linear systems

T.J.J. van den Boom [a,*] , B. De Schutter [a]

[a] *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands*
*Phone: +31-15-2784052/2785113*

**Abstract**

In this paper we consider the modelling and control of discrete event systems using switching max-plus-linear systems. In switching max-plus-linear systems we can switch between different modes of operation. In each mode the discrete event system is described by a max-plus-linear state space model with different system matrices for each mode. The switching allows us to change the structure of the system, to break synchronization and to change the order of events. We will give some examples of this type of systems.

We define the model predictive control design problem for this type of discrete event system, and we show that solving this problem in general leads to a mixed integer optimization problem.

*Key words:* discrete event systems, max-plus-linear systems, modeling, model predictive control

## 1 Introduction

The class of discrete event systems essentially consists of man-made systems that contain a finite number of resources (such as machines, communications channels, or processors) that are shared by several users (such as product types, information packets, or jobs) all of which contribute to the achievement of some common goal (the assembly of products, the end-to-end transmission of a set of information packets, or a parallel computation) (Baccelli et al., 1992). In general, models that describe the behavior of a discrete event system are nonlinear in conventional algebra. However, there is a class of discrete event systems — the max-plus-linear

* corresponding author
*Email address:* t.j.j.vandenboom@dcsc.tudelft.nl (T.J.J. van den Boom).

discrete event systems — that can be described by a model that is "linear" in the max-plus algebra (Baccelli et al., 1992; Cuninghame-Green, 1979), which has maximization and addition as its basic operations. The max-plus-linear discrete event systems can be characterized as the class of discrete event systems in which only synchronization and no concurrency or choice occurs.

In this paper we will consider discrete event systems that can switch between different modes of operation. In each mode the system is described by a max-plus-linear state space model with different system matrices for each mode. The switching changes the structure of the system, and so allows us to break synchronization and to change the order of events. We define a model predictive control design problem for such a system to optimize the system's behavior. In general this will lead to a mixed integer optimization problem, which can be solved using reliable solvers (Fletcher and Leyffer, 1998).

## 2 Max-plus algebra and switching max-plus-linear systems

In this section we present some basics in max-plus algebra and max-plus linear systems, and we will introduce the concept of switching max-plus linear systems.

### *2.1 Max-plus algebra and max-plus-linear systems*

In this section we define $\varepsilon = -\infty$ and $\mathbb{R}_{\max} = \mathbb{R} \cup \{\varepsilon\}$. The max-plus-algebraic addition ($\oplus$) and multiplication ($\otimes$) are defined as follows (Baccelli et al., 1992; Cuninghame-Green, 1979):

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

for numbers $x, y \in \mathbb{R}_{\max}$, and

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij})$$
$$[A \otimes C]_{ij} = \bigoplus_{k=1}^{n} a_{ik} \otimes c_{kj} = \max_{k=1,\ldots,n} (a_{ik} + c_{kj})$$

for matrices $A, B \in \mathbb{R}_{\max}^{m \times n}$ and $C \in \mathbb{R}_{\max}^{n \times p}$.

In Baccelli et al. (1992) it has been shown that discrete event systems in which there is synchronization but no concurrency or choice can be described by a model of the form

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

with $A \in \mathbb{R}_{\max}^{n \times n}$, $B \in \mathbb{R}_{\max}^{n \times m}$ and $C \in \mathbb{R}_{\max}^{m \times n}$, where $n$ is the number of states and $m$ the number of inputs. Systems that can be described by this model will be called max-plus-linear systems. The index $k$ is called the event counter. The matrices $A$, $B$, and $C$ usually consist of sums or maximizations of internal process times, transportation times, etc. For discrete event systems the state $x(k)$ typically contains the time instants at which the internal events occur for the $k$th time, and the input $u(k)$ contains the time instants at which the input events occur for the $k$th time.

### *2.2 Switching max-plus-linear systems*

In this paper we will consider discrete event systems that can switch between different modes of operation. Let the system be in mode $\ell(k) \in \{1, \ldots, n_{\mathrm{m}}\}$ for event step $k$, then the system is described by a max-plus-linear state space model

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

in which the matrices $A^{(\ell(k))}$, $B^{(\ell(k))}$ are the system matrices for mode $\ell(k)$. The switching allows us to model a change in the structure of the system, such as breaking a synchronization or changing the order of events. Note that each mode $\ell$ corresponds to a set of required synchronizations and an event order schedule, which leads to a model (2) with system matrices $(A^{(\ell(k))}, B^{(\ell(k))})$ for the $\ell$-th model (see also the examples in Section 3).

The moments of switching are determined by a switching mechanism. We define the switching variable $z(k)$, which may depend on the previous state $x(k-1)$, the previous mode $\ell(k-1)$, the input variable $u(k)$ and an (additional) control variable $v(k)$:

$$z(k) = \Phi(x(k-1), \ell(k-1), u(k), v(k)) \ \in \ \mathbb{R}_{\max}^{n_z} \quad . \tag{3}$$

We partition $\mathbb{R}_{\max}^{n_z}$ in $n_{\mathrm{m}}$ subsets $\mathscr{Z}^{(i)}$, $i = 1, \ldots, n_{\mathrm{m}}$. The mode $\ell(k)$ is now obtained by determining in which set $z(k)$ is for event step $k$. So if $z(k) \in \mathscr{Z}^{(i)}$, then $\ell(k) = i$. In some systems the switching mechanism will completely depend on the state $x(k-1)$ and input $u(k)$, in other examples $z(k)$ will be equal to $v(k)$ and so we can control the switching by choosing an appropriate $v(k)$.

Apart from the elegance of the switching MPL system, there is a clear motivation for studying this class of systems in more detail. Note that the basic idea of switching MPL systems in the event-driven domain is parallel to that of piece-wise-affine (PWA) systems in the time-driven domain. In the analysis of PWA systems, the properties of the linear subsystems are often employed to derive properties for the

PWA system (Chua and Deng, 1988; Leenaerts and van Bokhoven, 1998). Analogously, we will be able to use the properties of the MPL subsystems (i.e. the max-plus-algebraic eigenvalues, the eigenvectors, cycle time, the communication graph, etc.), for the analysis of the switching MPL system.

Note that to some extent the class of switching MPL systems is related to (max,+) automata (Gaubert, 1995), which can also be characterized as non-stationary autonomous max-plus-linear systems with finitely valued dynamics (i.e., systems of the form $x(k) = A(k) \otimes x(k-1)$, $y(k) = Cx(k)$ where $A(k)$ takes its values in a finite set $\{A^{(1)}, \ldots, A^{(N)}\}$. The main differences are that in the class of systems considered in this paper an additional input ($u(k)$) is present, and that the switching mechanism is specified completely and explicitly for the switching max-plus-linear systems, whereas for (max,+) automata this is not the case.

## 3 Examples of switching max-plus-linear systems

### *3.1 A scheduling problem*

Consider a production system with 6 machines $M_1$ to $M_6$. In order to produce a product, a job should be completed. A job consists of a number of operations that may be carried out on specific machines in a given order.

| Product | Recipe | | Machine 1 | Machine 2 | Machine 3 | Machine 4 | Machine 5 | Machine 6 |
|---|---|---|---|---|---|---|---|---|
| A | 1 | start | $\tau_1$ | $\tau_1+1$ | $\tau_1+2$ | | | |
| | | end | $\tau_1+2$ | $\tau_1+3$ | $\tau_1+3$ | | | |
| A | 2 | start | | | $\tau_2+2$ | $\tau_2+1$ | $\tau_2$ | |
| | | end | | | $\tau_2+3$ | $\tau_2+3$ | $\tau_2+2$ | |
| B | 3 | start | $\tau_3$ | $\tau_3+1$ | | | | $\tau_3+3$ |
| | | end | $\tau_3+2$ | $\tau_3+5$ | | | | $\tau_3+5$ |
| C | 4 | start | | | | | $\tau_4$ | $\tau_4+3$ |
| | | end | | | | | $\tau_4+4$ | $\tau_4+5$ |

Table 1
The nominal starting and stopping times for 4 types of recipes, where $\tau_i$, $i = 1, \ldots, 4$ is the starting time of a job using recipe $i$.

We can make 3 types of products. For product A we can use either recipe 1 or 2, for product B and C we use recipe 3 and 4, respectively. Table 1 gives the scheduling table with nominal starting and stopping times for the 4 types of recipes. For example, the production of product A following recipe 1 starts at time $\tau_1$ on machine 1, where it is processed for 2 time units. At time $\tau_1 + 1$ machine 2 starts to take

over the production, and also there the processing time is 2 time units. Machine 3 finalizes the product in time interval $[\tau_1+2, \tau_1+3]$.

Now let us consider the synchronization between jobs, and let $x_i(k)$ be the time instant at which machine $i$ is ready after $k$ jobs [1] . This means that if machine $i$ is idle during job $k$, we have $x_i(k) = x_i(k-1)$. For recipe 1 we have that the job can only start if the previous task in machine 1 has finished, so if we denote $\tau_1(k)$ as the starting time of job $k$ following recipe 1, we find $\tau_1(k) \geq x_1(k-1)$. Furthermore, from table 1 we see that job $k$ may start 1 time unit before the previous task in machine 2 has finished, so $\tau_1(k) \geq x_2(k-1)-1$. Finally, job $k$ may start 2 time units before the previous task in machine 3 has finished, so $\tau_1(k) \geq x_3(k-1)-2$. If we assume that the job starts as soon as possible we find $\tau_1(k) = \max(x_1(k-1), x_2(k-1)-1, x_3(k-1)-2)$, and thus

$$
\begin{aligned}
x_1(k) &= \tau_1(k)+2 = \max(x_1(k-1)+2, x_2(k-1)+1, x_3(k-1)) \\
x_2(k) &= \tau_1(k)+3 = \max(x_1(k-1)+3, x_2(k-1)+2, x_3(k-1)+1) \\
x_3(k) &= \tau_1(k)+3 = \max(x_1(k-1)+3, x_2(k-1)+2, x_3(k-1)+1) \\
x_4(k) &= x_4(k-1), \qquad x_5(k) = x_5(k-1), \qquad x_6(k) = x_6(k-1)
\end{aligned}
$$

For recipe 2 we have that the job may be started at time $\tau_2(k) = \max(x_3(k-1)-2, x_4(k-1)-1, x_5(k-1))$, and thus

$$
\begin{aligned}
x_3(k) &= \tau_2(k)+3 = \max(x_3(k-1)+1, x_4(k-1)+2, x_5(k-1)+3) \\
x_4(k) &= \tau_2(k)+3 = \max(x_3(k-1)+1, x_4(k-1)+2, x_5(k-1)+3) \\
x_5(k) &= \tau_2(k)+2 = \max(x_3(k-1), x_4(k-1)+1, x_5(k-1)+2) \\
x_1(k) &= x_1(k-1), \qquad x_2(k) = x_2(k-1), \qquad x_6(k) = x_6(k-1)
\end{aligned}
$$

For recipe 3 we have that the job may be started at time $\tau_3(k) = \max(x_1(k-1), x_2(k-1)-1, x_6(k-1)-3)$, and thus

$$
\begin{aligned}
x_1(k) &= \tau_3(k)+2 = \max(x_1(k-1)+2, x_2(k-1)+1, x_6(k-1)-1) \\
x_2(k) &= \tau_3(k)+5 = \max(x_1(k-1)+5, x_2(k-1)+4, x_6(k-1)+2) \\
x_6(k) &= \tau_3(k)+5 = \max(x_1(k-1)+5, x_2(k-1)+4, x_6(k-1)+2) \\
x_3(k) &= x_3(k-1), \qquad x_4(k) = x_4(k-1), \qquad x_5(k) = x_5(k-1)
\end{aligned}
$$

For recipe 4 we have that the job may be started at time $\tau_4(k) = \max(x_5(k-1), x_6(k-1)-3)$, and thus

[1] Note that this is different from most applications, where $x_i$ denotes the starting time of operation $i$.

$$
\begin{aligned}
x_5(k) &= \tau_4(k)+4 = \max(x_5(k-1)+4, x_6(k-1)+1)\\
x_6(k) &= \tau_4(k)+5 = \max(x_5(k-1)+5, x_6(k-1)+2)\\
x_1(k) &= x_1(k-1), \quad x_2(k) = x_2(k-1),\\
x_3(k) &= x_3(k-1), \quad x_4(k) = x_4(k-1)
\end{aligned}
$$

or in matrix form:

$$x(k) = A^{(\ell(k))} \otimes x(k-1)$$

where for recipe $i = 1,\ldots,4$ the system matrices $A^{(i)}$ are defined as

$$
A^{(1)} = \begin{bmatrix}
2 & 1 & 0 & \varepsilon & \varepsilon & \varepsilon \\
3 & 2 & 1 & \varepsilon & \varepsilon & \varepsilon \\
3 & 2 & 1 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0
\end{bmatrix}
\quad , \quad
A^{(2)} = \begin{bmatrix}
0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & 1 & 2 & 3 & \varepsilon \\
\varepsilon & \varepsilon & 1 & 2 & 3 & \varepsilon \\
\varepsilon & \varepsilon & 0 & 1 & 2 & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0
\end{bmatrix}
$$

$$
A^{(3)} = \begin{bmatrix}
2 & 1 & \varepsilon & \varepsilon & \varepsilon & -1 \\
5 & 4 & \varepsilon & \varepsilon & \varepsilon & 2 \\
\varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\
5 & 4 & \varepsilon & \varepsilon & \varepsilon & 2
\end{bmatrix}
\quad , \quad
A^{(4)} = \begin{bmatrix}
0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & 4 & 1 \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & 5 & 2
\end{bmatrix}
$$

Note that the $i,j$-th element of $A^{(\ell)}$ gives the relation between $x_i(k-1)$ and $x_j(k)$ for the $\ell$-th mode. We have the relation $x_j(k) \geq x_i(k-1) + a_{i,j}^{(\ell)}$. Recall that $\varepsilon = -\infty$. If $a_{i,j}^{(\ell)} = \varepsilon = -\infty$ it means that there is no synchronization between $x_i(k-1)$ and $x_j(k)$ (we have $x_j(k) \geq x_i(k-1) - \infty$, which is always true).

Figure 1 gives a graphical representation of the evolution of the scheduling system when

$$\Big(\ell(1),\ell(2),\ell(3),\ell(4)\Big) = (1,4,3,2)$$

and $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. From the figure it can be seen that we start the production at time 0 with recipe 1, which starts with machine 1 at time $t = 0$, as indicated by row A of Table 1. For $t = 1$ machine 2 starts and at time $t = 2$ machine 3 starts. Machine 1 has finished with recipe 1 at $t = 2$, machines 2 and 3 have finished with recipe 1 at $t = 3$. The recipe 4 is the next recipe in the schedule. For recipe 1 we do not use machines 5 and 6, necessary for recipe 4, and so we can immediately start at time $t = 0$ with machine 5, which runs until $t = 4$. Machine 6 will run from

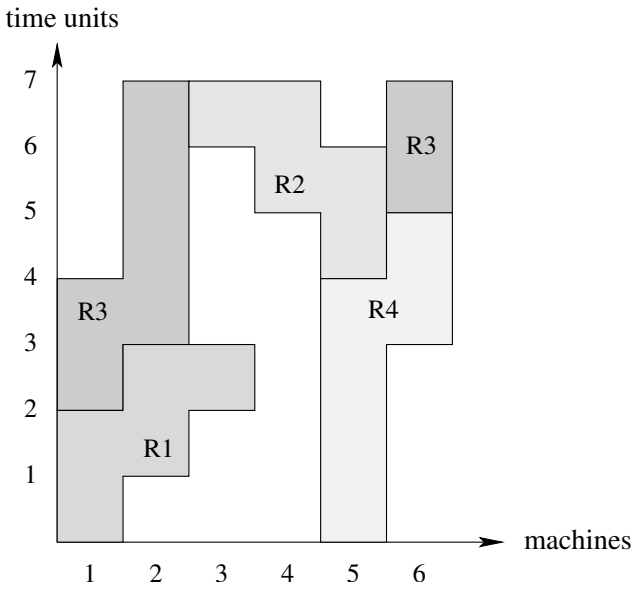


Figure 1. A scheduling system.

$t = 3$ to $t = 5$. Now recipe 3 is next. We can start recipe 3 after machines 1 and 2 are ready with recipe 1, and so at $t = 2$ machine 1 starts to operate for recipe 3, and machine 2 at time $t = 3$. Finally we will start recipe 2 at time $t = 4$ on machine 4. In the end, machine 1 is ready at time $t = 4$, machine 5 at time $t = 6$ and all other machines have finished at time $t = 7$. This means that after 4 jobs we find $x(4) = \begin{bmatrix} 4 & 7 & 7 & 7 & 6 & 7 \end{bmatrix}^T$.

### *3.2 Railway network*

Consider the railroad network of Figure 2 (see also van den Boom and De Schutter (2004b)). There are 4 stations in this railroad network (A, B, C and D) that are connected by 5 single tracks (1/7, 2/4, 3, 5, 9) and one double track (tracks 6 & 8). There are three trains available. The first train follows the route $D \to A \to B \to D$, the second train follows the route $A \to B \to C \to A$, and the third train follows the route $D \to A \to C \to D$. We assume that there exists a periodic timetable that schedules the earliest departure times of the trains. The period of the timetable is $T = 60$ minutes. So if a departure of a train from station B is scheduled at 5.30 a.m., then there is also scheduled a departure of a train from station B at 6.30 a.m., 7.30 a.m., and so on.

Each track of the railway network has a number and a train allocated to it. For the sake of simplicity we will say "(virtual) train $j$" to denote the (physical) train on a specific track. The number of tracks in the network is equal to 7, the number of physical trains in the network is equal to 3, and the number of virtual trains in the network is equal to 9. Let $x_j(k)$, $j = 1, \ldots, 9$ be the time instant at which train $j$

departs from its station for the $k$th time. Let $d_j(k)$ be the departure time for this train according to the time schedule, and let $a_j(k)$ be the transportation time for this train $j$.

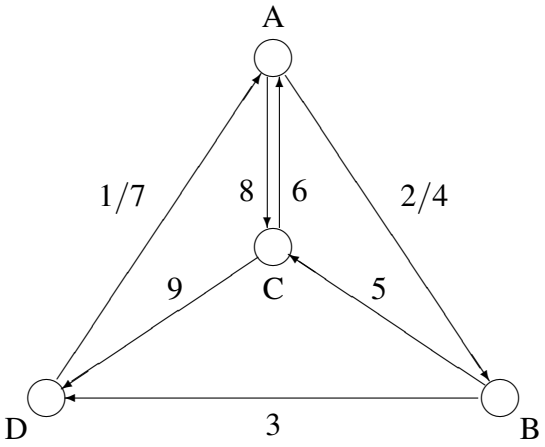


Figure 2. The railroad network of the example of Subsection 3.2.

Table 2 summarizes the information in connection with the nominal traveling times and the departure times. All the times are measured in minutes. The indicated departure times are the earliest departure times in the initial station of the track expressed in minutes after the hour. The first period starts at time $t = 0$. At the beginning of the first period the first physical train is in station D, the second physical train is in station A, and the third physical train is in station D.

| train | from -to | travel time | dep -arr | constraints |
|---|---|---|---|---|
| 1 | D-A | 12 | 00-12 | same train as $3^-$,connects to $9^-$,follows $7^-$ |
| 2 | A-B | 12 | 15-27 | same train as 1, connects to $6^-$, follows $4^-$ |
| 3 | B-D | 20 | 30-50 | same train as 2 |
| 4 | A-B | 12 | 19-31 | same train as $6^-$, follows 2, connects to 7 |
| 5 | B-C | 10 | 34-44 | same train as 4 |
| 6 | C-A | 25 | 47-12 | same train as 5 |
| 7 | D-A | 12 | 04-16 | same train as $9^-$, follows 1 |
| 8 | A-C | 25 | 19-44 | same train as 7 |
| 9 | C-D | 10 | 47-57 | same train as 8, connects to 5 |

Note: $3^-$ denotes train 3 in the previous cycle

Table 2
The nominal traveling times and the departure times for the railroad network.

The *continuity constraints* are that the trains on tracks 1, 2 and 3 are physically the same train, and the same holds for the trains on tracks 4, 5 and 6 and for the trains on tracks 7, 8 and 9.

*Connection constraints* are introduced to allow the passengers to change trains. In this network, train 1 has to wait for train 9 in the previous cycle with minimum connection time $c^{\min} = 3$. In the same way, train 2 waits for train 6 in the previous cycle, train 4 waits for train 7, and train 9 waits for train 5. The minimum stopping time of train $j$ at station $j$ to allow passenger to get off or on the train is fixed at $s^{\min} = 1$.

*Follow constraints* are introduced to guarantee sufficient separation time between two trains on the same track (moving in the same direction). In this network, train 4 is scheduled behind train 2 (train 4 follows train 2) with a minimum separation time $f^{\min} = 4$. In the same way, train 2 follows train 4 in the previous cycle, train 7 follows train 1, and train 1 follows train 7 in the previous cycle.

Each train departs as soon as all the connections are guaranteed (except for a connection when it is broken), the passengers have gotten the opportunity to change over and the earliest departure time indicated in the timetable has passed. We assume that in the first period all the trains depart according to schedule. The $j$-th state $x_j(k)$ is the time instant at which the train on track $j$ departs from the initial station of the track for the $k$th time.

Now we write down the equations that describe the evolution of the $x_j(k)$'s. First we consider the train on track 1 and we determine $x_1(k)$, the time instant at which this train departs from station D for the $k$th time. The train has to wait at least until the train has arrived in station D for the $(k-1)$th time [2] and the passengers have got the time to get out of the train so we have $x_1(k) \geq x_3(k-1) + a_3(k-1) + s^{\min}$. Furthermore, the train on track 1 has to wait for the passengers of the train on track 9 in the $(k-1)$th cycle, which arrives in station D at time instant $x_9(k-1) + a_9(k-1)$. The passengers have $c^{\min} = 3$ minutes to change trains. Further, the train on track 1 has to follow the train on track 7 in the previous cycle with a minimum separation time $f^{\min} = 4$. According to the timetable [3] the train on track 1 can only depart after time instant $00 + k\,60$. Hence, we have

[2] Under nominal operations the $k$th train on track 1 (e.g., the one that departs from station *D* at 10.00 a.m.) proceeds to the $(k-1)$th train on track 3 (which has departed from station *B* at 9.30 a.m.) and *not* to the $k$th train on track 3 (which will depart from station *B* at 10.30 a.m.).

[3] Note that in a undisturbed, well-defined time schedule the term $d_i(k)$ will be the largest. However, if due to unforeseen circumstances (an incident, a late departure, etc.) one of the trains has a delay, the corresponding term can become larger than the others, train $i$ will depart later than the scheduled departure time $d_i(k)$ and will therefore also be delayed.

$$
\begin{aligned}
x_1(k) &= \max(\, x_3(k-1) + a_3(k-1) + s^{\min}\,,\, x_7(k-1) + f^{\min}\,,\\
&\qquad\qquad x_9(k-1) + a_9(k-1) + c^{\min}\,, d_1(k)\,)\\
&= \max(\, x_3(k-1) + 21\,, x_7(k-1) + 4\,,\, x_9(k-1) + 13\,, k\,60\,)
\end{aligned}
$$

for $k = 1, 2, \ldots$ with $x_3(0) = x_9(0) = \varepsilon$.

Using a similar reasoning, we find that the other departure times are given by

$$
\begin{aligned}
x_2(k) &= \max(x_1(k) + 13, x_4(k-1) + 4, x_6(k-1) + 28, 15 + k\,60)\\
x_3(k) &= \max(x_2(k) + 13, 30 + k\,60)\\
x_4(k) &= \max(x_2(k) + 4, x_6(k-1) + 26, x_7(k) + 15, 19 + k\,60)\\
x_5(k) &= \max(x_4(k) + 13, 34 + k\,60)\\
x_6(k) &= \max(x_5(k) + 11, 47 + k\,60)\\
x_7(k) &= \max(x_9(k-1) + 11, x_1(k) + 4, 4 + k\,60)\\
x_8(k) &= \max(x_7(k) + 13, 19 + k\,60)\\
x_9(k) &= \max(x_8(k) + 26, x_5(k) + 13, 47 + k\,60)
\end{aligned}
$$

for $k = 1, 2, \ldots$ with $x_j(0) = \varepsilon$ for $j = 1, 2, \ldots, 9$.

Using successive substitution we can eliminate all right-hand terms with index $k$. By defining the appropriate matrix $A^{(1)}$ and by using the $(\otimes, \oplus)$-notation, we can rewrite the state equations as:

$$
x(k) = A^{(1)} \otimes x(k-1) \oplus d(k) \quad . \tag{4}
$$

In the nominal operation we have assumed that some trains should give pre-defined connections to other trains, and the order of trains on the same track is fixed. However, if one of the preceding trains has a too large delay, then it is sometimes better — from a global performance viewpoint — to let a connecting train depart anyway or to change the departure order on a specific track. This is done in order to prevent an accumulation of delays in the network. In this paper we consider the switching between different operation modes, where each mode corresponds to a different set of pre-defined or broken connections and a specific order of train departures. We allow the system to switch between different modes, allowing us to break train connections and to change the order of trains. Note that any broken connection or change of train order leads to a new model, similar to the nominal equation (4), but now with adapted system matrix $A^{(\ell(k))}$ for the $\ell$-th model. We have the following system equation for the perturbed operation for $\ell(k) \in \{2, \ldots, n_{\mathrm{m}}\}$:

$$
x(k) = A^{(\ell(k))} \otimes x(k-1) \oplus d(k) \quad . \tag{5}
$$

In this railway network the switching variable $z(k)$ is equal to the control vector $v(k)$, and each entry of $v(k)$ corresponds to a specific control action, so a specific (scheduled) synchronization or specific (scheduled) event order. We assume $v(k)$ to be binary, where $v_i(k) = 0$ corresponds to the nominal case, and $v_i(k) = 1$ to a perturbed case (a synchronization is broken or the order of two events is switched).

Each combination $v_1(k),\ldots,v_m(k)$ corresponds to a fixed routing schedule with a specific train order and specific connections.

### *3.3 Production system with concurrency*

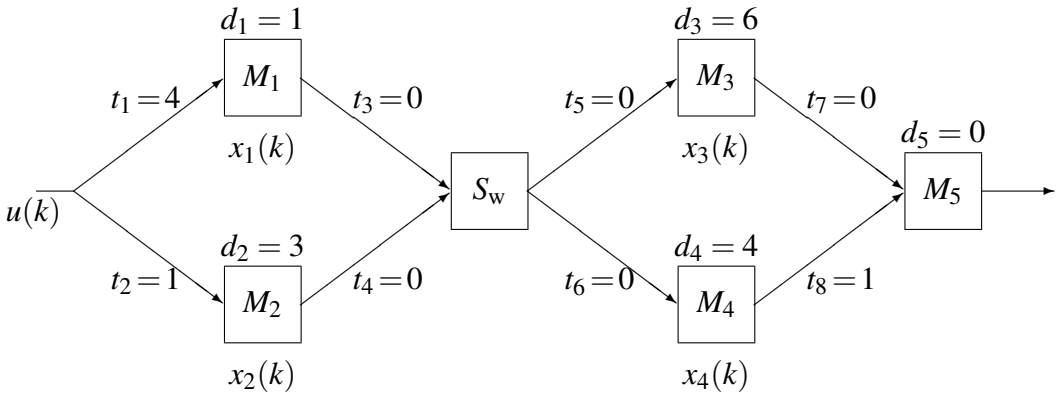


Figure 3. A production system.

Consider the production system of Figure 3. This system consists of five machines $M_1$, $M_2$, $M_3$, $M_4$ and $M_5$. The raw material is fed to machine $M_1$ and $M_2$, where preprocessing is done. Both intermediate products now have to be finished in either unit $M_3$ and $M_4$, which basically perform the same task, but the processing time of $M_3$ is longer than $M_4$. Therefore, the products coming from machine $M_1$ and $M_2$ are directed to a switching device $S_w$, that feeds the first product in the $k$th cycle to the slower machine $M_3$ and the second product to the faster machine $M_4$. Finally, the products are assembled (instantaneously [4]) in machine $M_5$ and become available. 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 products are available, and as soon as the machine is idle (i.e., the previous batch has been finished and has left the machine). We define $u(k)$ as the time instant at which the system is fed for the $k$th time, and $x_i(k)$ as the time instant at which machine $i$ starts for the $k$th time. The variable $t_j$ for $j = 1,\ldots,8$ is the transportation time, and $d_i$ for $i = 1,\ldots,5$ is the processing time on machine $i$. The system equations for $x_1$ and $x_2$ are given by

$$x_1(k) = \max(x_1(k-1)+d_1, u(k)+t_1) \tag{6}$$
$$x_2(k) = \max(x_2(k-1)+d_2, u(k)+t_2) \tag{7}$$

Equation (6) can be explained as follows: machine 1 will start with job $k$ when

i) the previous job is finished, indicated by $x_1(k-1)+d_1$ (=the start of the previous job $x_1(k-1)$ + the production time $d_1$), and
ii) the raw material has arrived at the machine at time $u(k)+t_1$ (=the time the raw material is put into the system $u(k)$ + the transportation time $t_1$).

[4] i.e. with a negligible processing time.

For Equation (7) we have that machine 2 will start when

i) the previous job is finished at time $x_2(k-1)+d_2$ (=the start of the previous job $x_2(k-1)$ + the production time $d_2$) and
ii) the raw material has arrived at the machine $u(k)+t_2$ (=the time the raw material is put into the system $u(k)$ + the transportation time $t_2$).

If $x_1(k)+d_1 \leq x_2(k)+d_2$ (i.e. machine 1 finishes first, machine 2 finishes later), the product of $M_1$ will be directed to $M_3$ and the product of $M_2$ will be directed to $M_4$. Machine 3 will start when

i) the previous job is finished $x_3(k-1)+d_3$ (=the start of the previous job $x_3(k-1)$ + the production time $d_3$), and
ii) the intermediate product has arrived from machine 1: $x_1(k)+d_1$ (=the start of the job $x_1(k)$ + the production time $d_1$ + transportation time $t_3=0$).

So

$$x_3(k) = \max(x_1(k)+d_1, x_3(k-1)+d_3).$$

By substitution of (6) we obtain:

$$x_3(k) = \max(x_1(k-1)+2d_1, x_3(k-1)+d_3, u(k)+d_1+t_1).$$

In a similar way we derive

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

Note that $t_3,\ldots,t_7=0$, and therefore they do not appear in the equations. For this first mode ($x_1(k)+d_1 \leq x_2(k)+d_2$, so machine 1 finishes first, machine 2 finishes later) we obtain the system matrices

$$A^{(1)} = \begin{bmatrix} 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ 2 & \varepsilon & 6 & \varepsilon & \varepsilon \\ \varepsilon & 6 & \varepsilon & 4 & \varepsilon \\ 8 & 11 & 12 & 9 & \varepsilon \end{bmatrix} \quad B^{(1)} = \begin{bmatrix} 4 \\ 1 \\ 5 \\ 4 \\ 11 \end{bmatrix}.$$

Similarly, for the second mode ($x_1(k)+d_1 > x_2(k)+d_2$, so machine 2 finishes first, machine 1 finishes later) we obtain the system matrices

$$A^{(2)} = \begin{bmatrix} 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 6 & 6 & \varepsilon & \varepsilon \\ 2 & \varepsilon & \varepsilon & 4 & \varepsilon \\ 7 & 12 & 12 & 9 & \varepsilon \end{bmatrix} \quad B^{(2)} = \begin{bmatrix} 4 \\ 1 \\ 4 \\ 5 \\ 10 \end{bmatrix}.$$

To decide the switching mechanism, we define the switching variable

$$\begin{aligned} \begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} &= \begin{bmatrix} x_1(k)+d_1 \\ x_2(k)+d_2 \end{bmatrix} \\ &= \begin{bmatrix} \max(x_1(k-1)+2d_1, u(k)+d_1+t_1) \\ \max(x_2(k-1)+2d_2, u(k)+d_2+t_2) \end{bmatrix} \\ &= \begin{bmatrix} \max(x_1(k-1)+2, u(k)+5) \\ \max(x_2(k-1)+6, u(k)+4) \end{bmatrix}, \end{aligned}$$

and the sets

$$\begin{aligned} \mathscr{Z}^{(1)} &= \{z \in \mathbb{R}^2_{\max} | z_1 \leq z_2\}, \\ \mathscr{Z}^{(2)} &= \{z \in \mathbb{R}^2_{\max} | z_1 > z_2\}. \end{aligned}$$

Note that $z_1(k)$ and $z_2(k)$ are the time instants at which machines 1 and 2, respectively, finish their product in cycle $k$. With that in mind, it is clear that mode 1 corresponds to "machine 1 finishes first, machine 2 finishes later" ($z_1 \leq z_2$) and mode 2 corresponds to "machine 2 finishes first, machine 1 finishes later" ($z_1 > z_2$). Now the state space equation of our system is given by (2).

## 4 The model predictive control problem

Consider the switching max-plus-linear model (2)–(3). We have two possible input signals, $v(k)$ and $u(k)$. Just as in conventional Model Predictive Control (MPC) (Maciejowski, 2002) we define the input sequences $\tilde{u}(k) = \left[u^T(k), \ldots, u^T(k+N_\mathrm{p}-1)\right]^T$

and $\tilde{v}(k) = \left[ v^T(k), \ldots, v^T(k+N_\mathrm{p}-1) \right]^T$ where $N_\mathrm{p}$ is the prediction horizon (so it determines how many cycles we look ahead in our control law design). Let $\mathscr{V}(k)$ and $\mathscr{U}(k)$ be the sets of possible future control sequences $\tilde{v}(k)$ and $\tilde{u}(k)$, respectively. Sometimes values are predefined (e.g. in the railway system $u(k) = d(k)$) or not applicable (e.g. in the production system $v(k)$ is not used). Often $v(k)$ is assumed to be binary, and each entry corresponds to a specific control action (e.g. a specific scheduled synchronization or specific scheduled event order). In other cases $v(k)$ is an integer (e.g. for the scheduling problem we have $v(k) = \ell(k)$).

We now aim at computing the optimal $\tilde{u}(k)$ and $\tilde{v}(k)$ that minimize a cost criterion $J(k)$, possibly subject to linear constraints on the inputs and the states. The cost criterion reflects the input and output cost functions ($J_\mathrm{in}$ and $J_\mathrm{out}$, respectively) in the event period $[k, k+N_\mathrm{p}-1]$:

$$J(k) = J_\mathrm{out}(k) + \lambda J_\mathrm{in}(k) \quad , \tag{8}$$

where the weight $\lambda \geq 0$ is a tuning parameter, chosen by the user. The output cost function is usually chosen as

$$J_\mathrm{out}(k) = \sum_{j=0}^{N_\mathrm{p}-1} \| e(k+j) \| \quad ,$$

where $\| \cdot \|$ is an appropriate norm (usually the two-norm, the one-norm or the infinity-norm), and $e$ is the due date error (e.g. for the production system the due date error of a product is given by $e_i(k) = \max(y_i(k) - r_i(k), 0)$, where $r(k)$ is the due date of the product, and for the railway system the due date error is equal to the delay of a train, so $e_i(k) = \max(x_i(k) - d_i(k), 0)$). The input cost function consists of two parts, $J_\mathrm{in} = J_{\mathrm{in},u} + J_{\mathrm{in},v}$. The first part $J_{\mathrm{in},u}$ depends on $\tilde{u}(k)$ and is usually chosen as

$$J_{\mathrm{in},u}(k) = -\sum_{j=0}^{N_\mathrm{p}-1} \| u(k+j) \| \quad ,$$

(see also De Schutter and van den Boom (2001)). The second part $J_{\mathrm{in},v}$ is a function of $\tilde{v}(k)$. For different applications, $J_{\mathrm{in},v}$ will have different appearances (e.g. for the production system $J_{\mathrm{in},v}(k) = 0$ and for the railway system we choose $J_{\mathrm{in},v}(k) = \sum_{j=0}^{N_\mathrm{p}-1} \sum_{i=1}^{n_v} \alpha_i v_i(k+j)$, where $\alpha_i \geq 0$ are weights to punish any action corresponding to the $i$th variable).
The parameter $\lambda$ makes a trade-off between the output cost function and the input cost function. For $\lambda = 0$ the input cost function is not taken into account, for $\lambda \to \infty$ the output function will not be taken into account. The time interval $[1, N_\mathrm{p}]$ should contain the crucial dynamics of the process, and important information of the due date sequence.

Since the input signals $u(k)$ correspond to consecutive event occurrence times, we have the additional condition for $j = 0, \ldots, N_\mathrm{p}-1$:

$$\Delta u(k+j) = u(k+j) - u(k+j-1) \geq 0 \, , \tag{9}$$

which means that the start of the $(k+j)$th event (i.e. at time $u(k+j)$) will always be later than the start of the $(k+j-1)$th event (i.e. at time $u(k+j-1)$). Furthermore, in order to reduce the number of decision variables and the corresponding computational complexity we introduce a control horizon $N_c$ ($\leq N_p$) and we impose the additional condition that the input rate should be constant from the point $k+N_c-1$ on, so

$$\Delta u(k+m) = \Delta u(k+N_c-1)\,, \tag{10}$$

for $m = N_c,\ldots,N_p-1$. So, instead of allowing the future control actions to be "free", the increments of $\Delta u(k)$ are assumed to become constant after event step $(k+N_c-1)$, i.e. we have a constant feeding rate. The parameter control horizon $N_c$ can be chosen between 1 and $N_p$. The same procedure is often followed for the control variable $v(k)$, which will be assumed to be constant beyond control horizon $N_c$. This results in the constraint $v(k+m) = v(k+N_c-1)$ for $m = N_c,\ldots,N_p-1$, or equivalently

$$\Delta v(k+m) = v(k+m) - v(k+m-1) = 0\,, \tag{11}$$

for $m = N_c,\ldots,N_p-1$. Now the MPC control problem for event step $k$ can be defined as:

$$\min_{\{\tilde{u}(k)\in\mathscr{U},\tilde{v}(k)\in\mathscr{V}(k)\}} J(k) \tag{12}$$

subject to

$$x(k+j) = A^{(\ell(k))}(k)\otimes x(k+j-1)\oplus B^{(\ell(k))}(k)\otimes u(k+j), \quad \text{for } j=0,\ldots,N_p-1, \tag{13}$$

$$\Phi(k+j-1),\ell(k+j-1),u(k+j),v(k+j))\in\mathscr{Z}^{(\ell(k+j))}, \quad \text{for } j=0,\ldots,N_p-1, \tag{14}$$

$$\Delta u(k+j)\geq 0, \quad \text{for } j=0,\ldots,N_p-1, \tag{15}$$

$$\Delta v(k+m) = 0, \quad \text{for } m=N_c,\ldots,N_p-1, \tag{16}$$

$$\Delta u(k+m) - \Delta u(k+N_c-1) = 0, \quad \text{for } m=N_c,\ldots,N_p-1, \tag{17}$$

$$A_c(k)\tilde{u}(k) + B_c(k)\tilde{y}(k) \leq c_c(k) \tag{18}$$

where (12) is the cost function we want to minimize, (13) represents the system dynamics we have to take into account, (14) represents the switching mechanism, (15) guarantees a non-decreasing input sequence (cf. (9)), (16) and (17) introduce the control horizon for the signals $\Delta u$ and $v$, respectively (cf. (10) and (11)), and (18) represents possible additional linear constraints on the inputs and the outputs. Note that for $N_c = N_p$, equations (16) and (17) vanish.

MPC uses a receding horizon principle. This means that after computation of the optimal future control sequences $\tilde{u}(k)$ and $\tilde{v}(k)$, only the first control samples $u(k)$

and $v(k)$ will be implemented, subsequently the horizon is shifted one sample, and the optimization is restarted with new information of the measurements.

In principle we have all elements to solve the receding horizon control problem (12)–(18). In general we will have an optimal control problem with both real parameters and integer parameters. Sometimes (e.g. the production system) the problem can be recast as an Extended Linear Complementary Problem (ELCP) and can be solved efficiently (De Schutter and van den Boom, 2002). If the optimization is over a binary valued $v(k)$ (e.g. the railway problem) we obtain an integer optimization problem (without any real valued variables), which can be solved using genetic algorithms (Davis, 1991), tabu search (Glover and Laguna, 1997), or a branch-and-bound method (Cordier et al., 1999). In some particular cases the problem can be recast as a Mixed Integer Linear Programming (MILP) or a Mixed Integer Quadratic Programming (MIQP), for which reliable algorithms are available (Bemporad and Morari, 1999; Fletcher and Leyffer, 1998).

The application of the derived controller design method to a railway network is given in van den Boom and De Schutter (2004b).

*Timing*

MPC for (switching) MPL systems is different from conventional MPC in the sense that the event counter $k$ is not directly related to a specific time (van den Boom and De Schutter, 2002). So far we have assumed that $x(k-1)$ is available when we optimize over the future control sequence. However, not all components of $x(k-1)$ are known at the same time instant [5] . Therefore, we will now present a method to address the timing issues of the controller.

We consider the case of full state information [6] . Let $[x_{\text{true}}(k-1)]_i$ be the measured (true) occurrence time of the $(k-1)$th occurrence of internal event $i$, and let $[x_{\text{est}}(k-1|t)]_i$ be an estimation of the $(k-1)$th occurrence time of this event at time $t$. The estimation can be done using the following procedure: Let $m(t)$ be the smallest integer such that $[x_{\text{true}}(k-m(t))]_i < t$ for all $i = 1,\ldots,n$. Hence, $[x_{\text{true}}(k-m(t))]_i$ is completely known at time $t$. If we define $x_{\text{est}}(k-m(t)|t) = x_{\text{true}}(k-m(t))$, we can reconstruct the unknown state components using the recursion

$$x_{\text{est}}(k-j|t) = A^{(\ell(k|t))} \otimes x_{\text{est}}(k-j-1|t) \oplus B^{(\ell(k|t))} \otimes u(k-j|t)$$
$$\Phi(x_{\text{est}}(k-j-1|t), \ell(k-j-1|t), u(k-j|t), v(k-j|t)) \in \mathscr{Z}^{(\ell(k-j|t))}$$

[5] Recall that $x(k-1)$ contains the time instants at which the internal activities or processes of the system start for the $(k-1)$th time.

[6] Since the components of $x$ correspond to event times, they are in general easy to measure. Also note that measurements of occurrence times of events are in general not as susceptible to noise and measurement errors as measurements of continuous-time signals involving variables such as temperature, speed, pressure, etc.

for $j = 1, \ldots, m(t)-1$, where for the components of $u(k-m(t)+j|t)$ that are less than $t$ we take the actually applied input times, and for the other components we take the computed values. Finally, the value of the state $x(k)$ that can be used to compute the MPC controller at time $t$ is given by $x(k-1|t)$ with components $[x(k-1|t)]_i$ for $i = 1, \ldots, n$ such that

$$[x(k-1|t)]_i = \begin{cases} [x_{\text{true}}(k-1)]_i & \text{if } [x_{\text{true}}(k-1)]_i \text{ is known at time } t \\ & \qquad (\text{so } [x_{\text{true}}(k-1)]_i \leq t) \\ [x_{\text{est}}(k-1|t)]_i & \text{otherwise.} \end{cases}$$

Finally, before the implementation of the controller can be done, one has to determine at what time instants a new optimization should be done. In principle, the appropriate input sequences $\tilde{u}(k)$ and $\tilde{v}(k)$ should be recomputed, as soon as a new measurement of state $[x_{\text{true}}(k-1)]_i$ comes available. If the measured $[x_{\text{true}}(k-1)]_i$ is equal to the estimated $[x_{\text{est}}(k-1|t)]_i$, an optimization is superfluous and the already computed input sequences will be optimal.

If we have multiple inputs, (and so $u(k)$ is a vector), we will implement $u_i(k)$ as soon as it is equal to time $t$. Let $u_i(k) = t_0$ and let us consider an optimization of $u(k)$ for $t > t_0$. An event in the past cannot be changed any more, and so we will do the optimization of $\tilde{u}(k)$ subject to an an additional equality constraint $u_i(k) = t_0$.

## 5 Examples of controlled switching max-plus-linear systems

### *5.1 A scheduling problem*

Consider the scheduling example from Section 3.1. The control variable in this example is equal to the chosen recipe, so $v(k) = \ell(k) \in \{1,2,3,4\}$ is an integer variable ($u$ is not applicable in this example).

Let $n_{\text{X}}(k)$, X$\in$ {A,B,C}, denote the number of products X produced until cycle $k$. Then $n_{\text{A}}(k)$, $n_{\text{B}}(k)$, and $n_{\text{C}}(k)$ can be written as

$$n_{\text{A}}(k) = \sum_{j=-\infty}^{k} \delta\Big(\ell(j)-1\Big) + \delta\Big(\ell(j)-2\Big)$$

$$n_{\text{B}}(k) = \sum_{j=-\infty}^{k} \delta\Big(\ell(j)-3\Big)$$

$$n_{\text{C}}(k) = \sum_{j=-\infty}^{k} \delta\Big(\ell(j)-4\Big)$$

where $\delta(n)$ is given by

$$\delta(n) = \begin{cases} 1 \text{ for } n = 0 \\ 0 \text{ elsewhere} \end{cases}$$

The goal in this scheduling example is to produce products according to some production rate

$$\Delta g(k) = \begin{bmatrix} \Delta g_{\mathrm{A}}(k) & \Delta g_{\mathrm{B}}(k) & \Delta g_{\mathrm{C}}(k) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 & 0.25 \end{bmatrix}.$$

which means that (on the average) we aim to deliver a product $A$ in two production steps, and a product $B$ and a product $C$ in four production steps.

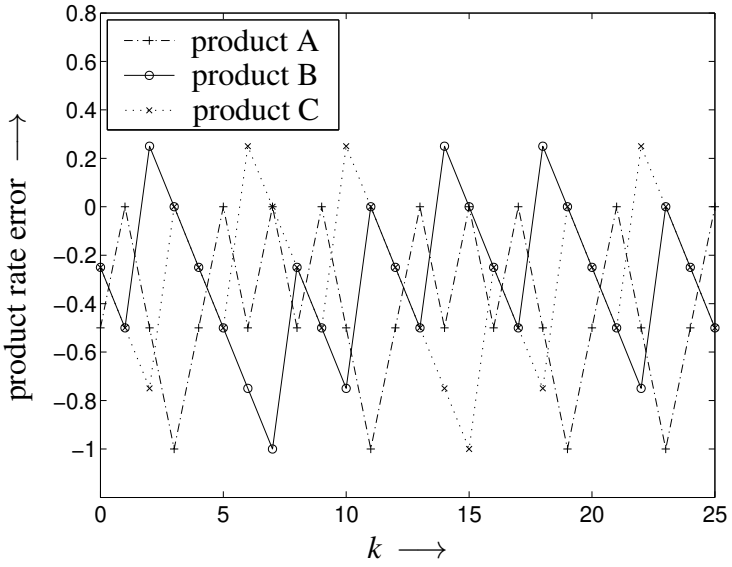


Figure 4. Product rate error for a scheduling system

The desired number of products X, that is in stock after cycle $k$, is equal to

$$g_{\mathrm{X}}(k) = \sum_{j=-\infty}^{k} \Delta g_{\mathrm{X}}(j), \quad \mathrm{X} \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}.$$

Our cost criterion is chosen as

$$\begin{aligned} J(k) = \max(x_i(k+N_{\mathrm{p}})) + \lambda \sum_{j=1}^{N_{\mathrm{p}}} \Big( & |n_{\mathrm{A}}(k+j) - g_{\mathrm{A}}(k+j)| \\ & + |n_{\mathrm{B}}(k+j) - g_{\mathrm{B}}(k+j)| + |n_{\mathrm{C}}(k+j) - g_{\mathrm{C}}(k+j)| \Big) \end{aligned} \tag{19}$$

where the first term $(\max(x_i(k+N_{\mathrm{p}})))$ represents the total production time for jobs $1, \ldots, N_{\mathrm{p}}$, and the last three terms $\sum_{j=1}^{N_{\mathrm{p}}} |n_{\mathrm{X}}(k+j) - g_{\mathrm{X}}(k+j)|$, with $X = A, B, C$ each

represent the production rate error for products $A$, $B$, and $C$, respectively. Recall Note that $n_{\mathrm{X}}(k)$ is the number of products $X$ that are actually produced after cycle $k$, and $g_{\mathrm{X}}(k)$ is the number of products $X$ that should have been produced after cycle $k$. The parameter $\lambda$ makes a trade-off between a minimum total production time and the sum of the errors in the production rate.

In this example we now minimize $J(k)$ for $\lambda = 25$, $N_{\mathrm{p}} = N_{\mathrm{c}} = 20$, and we do a simulation for $k = 0, \ldots, 80$. To account for variations in the production time, we add a noise-term (normal distributed with mean zero and variance one) to the real production time of every machine. The input signal is optimized using a genetic algorithm (Davis, 1991), and we obtain the optimal sequence $\ell(k)$, $k > 0$.

In Figure 4 the product rate error is given for each $k$ over the first 25 cycles. We see that the product rate error varies between $-0.75$ and 0.5. This is due to the fact that the increment of $g_{\mathrm{X}}$ is 0.5 or 0.25, where the increment of $n_{\mathrm{X}}$ can only be 0 or 1.

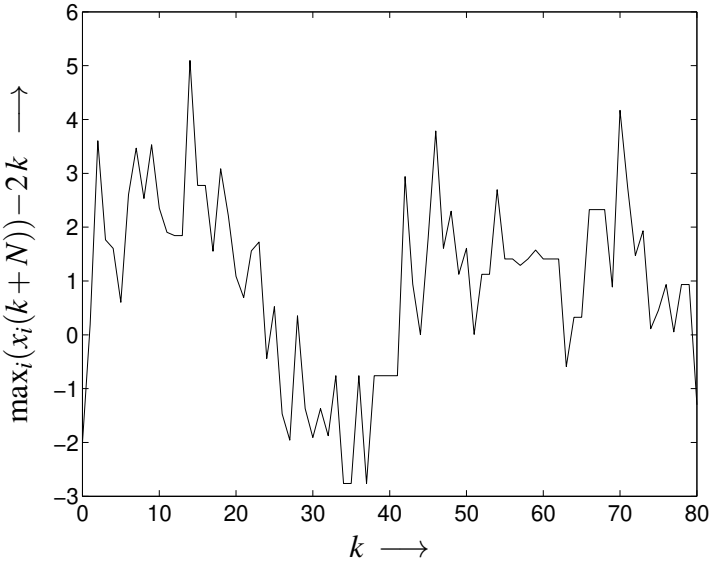


Figure 5. Variation of the production time for a scheduling system

The average predicted final production time over 80 cycles is about 2.0 time units/cycle. Therefore Figure 5 gives the deviation between the predicted final production time and the average production time $(\max_i(x_i(k+N)) - 2k)$ for each cycle $k$. From the figure we see that the production time is varying about the average value. This variation is due to the variation in the production times of the units. In the noiseless case the average production time will decrease to 1.5, which means that the average job time (4 cycles) is equal to 6. From Figure 1 it can easily be derived that for the noiseless case a possible realization of the optimal sequence is given by $\ell(k)$ with

$$\ell(4n+1) = 1, \quad \ell(4n+2) = 4, \quad \ell(4n+3) = 3, \quad \ell(4n+4) = 2, \quad \text{for } n \in \mathbb{Z}^+.$$

Using this sequence leads to a schedule where machine 2 is never idle and works constantly. The two operation times, related to recipe 1 and 3, together sum up to the total job time 6, and so to reduce the total job time we need to remove either recipe 1 or recipe 3 from the schedule. Recipe 3 cannot be removed because it is related to product 3, which can only be produced using machine 2. This means that we should replace operation recipe 1 by recipe 2, both related to the manufacturing of a product 1. However, in that case we will need machine 5 during 3 recipes (recipe 4 and twice recipe 2) giving a job time of at least 8. The same argument can be used for machine 5, and we will conclude that the given sequence is indeed optimal.

### *5.2 Railway network*

Consider the railroad network of Section 3.2. We assume $u(k) = d(k)$ is fixed and $v(k)$ is a binary parameter vector. Consider the cost function

$$J(k) = \sum_{j=0}^{N_\mathrm{p}-1} \left( \sum_{i=1}^{n} x_i(k+j) - d_i(k+j) + \sum_{l=1}^{n_v} \lambda_l \, v_l(k+j) \right) \tag{20}$$

In this function $x_i(k+j) - d_i(k+j)$ is the delay of train $i$ in cycle $j$. Note that the departure time $x_i(k+j)$ is always larger or equal to the scheduled time according to the time table $d_i(k+j)$, because a train is never allowed to leave too early. The first term is the sum of all delays in the next $N_\mathrm{p}$ cycles. The second term reflects the costs $\lambda_l$ we make by our control actions $v_l(k+j)$ (breaking connections or changing the order of trains). We solve the optimal control problem of minimizing the cost function (20) subject to constraints (13)–(18), where $N_\mathrm{p}$ is chosen sufficiently large.

This results in an integer optimization problem. We introduce a perturbation at time $t = 0$ and let train 3, 6 and 9 arrive with a delay with 43, 20 and 32 minutes respectively. We choose $\lambda_l = 500$ for inputs $v_l$ related to connections and $\lambda_l = 10$ for the other inputs. The input signal is optimized with a branch-and-bound algorithm and we obtain the optimal sequence (for more details, see van den Boom and De Schutter (2004b)). To find a good initial guess for the branch-and-bound algorithm we first solve an easier problem, in which we parameterize the input signal $v(k+j)$ with two threshold values $(\tau, \phi)$ and a decision mechanism. We use the thresholds on (expected) delays to decide whether a connection should be broken or train orders should be switched (see also van den Boom and De Schutter (2004b)). Loosely speaking, the decision mechanism breaks the connection between two trains if the expected delay exceeds a threshold value of $\tau$, and it changes the order of two trains moving over the same track if the expected delay exceeds a threshold value of $\phi$. Optimizing the threshold values $(\tau, \phi)$ leads to a real-valued two-dimensional optimization problem, that we can solve using a Sequential Quadratic Programming (SQP) algorithm. We find a minimum cost function $J_\mathrm{dec} = 4.09 \cdot 10^6$. The optimal

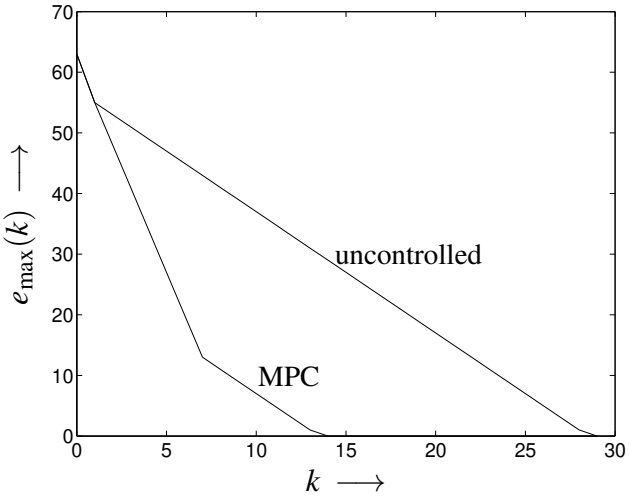


Figure 6. Maximum delay for an uncontrolled railway system and a railway system using MPC

threshold values, together with the decision mechanism, lead to a sequence $v(k+j)$ that is used as an initial value for the branch-and-bound algorithm. We find a minimum cost function $J_{\text{b\&b}} = 3.89 \cdot 10^6$.

In Figure 6 the maximum delay $e_{\max}(k) = \max_i(x_i(k) - d_i(k))$ in each cycle $k$ is given for both the uncontrolled case (so $v(k+j) = 0$ for all $j > 0$) and for the MPC case (with $v^*(k+j)$). We see that the delay in the MPC case decays much faster than the uncontrolled case.

### *5.3 Production system with concurrency*

Consider the production system of Section 3.3. Input $u(k)$ is the control variable in this example ($v$ is not applicable here). The cost criterion is now given by

$$J(k) = \sum_{j=0}^{N_{\text{p}}-1} \max(x_5(k+j) - r(k+j), 0) - \lambda u(k+j) \tag{21}$$

where $r(k)$ is the due date of the product, the prediction horizon $N_{\text{p}} = 3$ and $\lambda = 0.1$. In this example we will not use a control horizon, i.e. $N_{\text{c}} = N_{\text{p}}$.

From here we could solve the MPC control problem in the way of Section 4, and we would end up with a mixed integer optimization problem. However, the system in this example belongs to the class of max-min-plus (MMP) systems, which is a subclass of the switching MPL systems. As we will show in this subsection for this important subclass of MMP systems the MPC optimization problem can be reduced to solving a set of linear programming problems, and so the computational burden is reduced dramatically.

First we will discuss the class of max-min-plus (MMP) systems.

**Definition 1** *An MMP expression $f$ of the variables $w_1, \ldots, w_n$ is defined by the grammar* [7]

$$f := w_i + \alpha \,|\, \max(f_k, f_l) \,|\, \min(f_k, f_l) \tag{22}$$

*with $i \in 1, \ldots, n$, $\alpha \in \mathbb{R}$, and where $f_k$, $f_l$ are again MMP expressions.*

Consider now a system that can be described by

$$x(k) = \mathscr{M}(x(k-1), u(k)) \tag{23}$$

where $\mathscr{M}$ is an MMP expression in terms of the components of $x(k-1)$ and $u(k)$. Such a system will be called an MMP system [8].

**Lemma 1** *The class of MMP systems is a subclass of the class of switching MPL systems.*

The proof of Lemma 1 is in the appendix.

We can now use the results of De Schutter and van den Boom (2002, 2004) to solve the MPC problem for the production system. Define $z_{\min}(k)$ and $z_{\max}(k)$ as:

$$\begin{aligned}
z_{\min}(k) &= \min(z_1(k), z_2(k)) \\
&= \min\Big(\max(x_1(k-1)+2, u(k)+5), \max(x_2(k-1)+6, u(k)+4)\Big) \\
z_{\max}(k) &= \max(z_1(k), z_2(k)) \\
&= \max\Big(\max(x_1(k-1)+2, u(k)+5), \max(x_2(k-1)+6, u(k)+4)\Big) \\
&= \max\Big(x_1(k-1)+2, x_2(k-1)+6, u(k)+5\Big),
\end{aligned}$$

Note that $z_{\min}(k) = \min(z_1(k), z_2(k))$ can be interpreted as the time instant that the first of the machines 1 or 2 is ready in cycle $k$, whereas $z_{\max}(k) = \max(z_1(k), z_2(k))$ can be interpreted as the time instant that both machines 1 and 2 are ready in cycle $k$. With these definitions of $z_{\min}$ and $z_{\max}$, we obtain

[7] The symbol | stands for OR and the definition is recursive.

[8] An MMP system is a special case of the more general max-min-plus-scaling (MMPS) system as defined in De Schutter and van den Boom (2002, 2004). Note that in for MMP systems there is no scaling necessary, because $x$ and $u$ denote time-instants and time does not stretch or shrink

$$
\begin{aligned}
x_3(k) &= \max(z_{\min}(k), x_3(k-1)+6) \\
&= \max(\min(\max(x_1(k-1)+2, u(k)+5), \max(x_2(k-1)+6, u(k)+4)), \\
&\quad x_3(k-1)+6) \\
x_4(k) &= \max(z_{\max}(k), x_4(k-1)+4) \\
&= \max(x_1(k-1)+2, x_2(k-1)+6, x_4(k-1)+4, u(k)+5) \\
x_5(k) &= \max(x_3(k)+6, x_4(k)+5) \\
&= \max(\min(\max(x_1(k-1)+8, u(k)+11), \max(x_2(k-1)+11, u(k)+10)), \\
&\quad x_3(k-1)+12, x_1(k-1)+7, x_2(k-1)+11, x_4(k-1)+9, u(k)+10).
\end{aligned}
$$

Using successive substitution we can easily derive expressions for $x_5(k+1)$, $x_5(k+2)$ and $J(k)$. These functions are max-min-plus (MMP) functions, and the optimization problem of minimizing $J(k)$ subject to (9) or equivalently (15) can now be solved using a set of linear programming problems (De Schutter and van den Boom, 2002, 2004). In our case we obtain 64 linear programming (LP) problems.

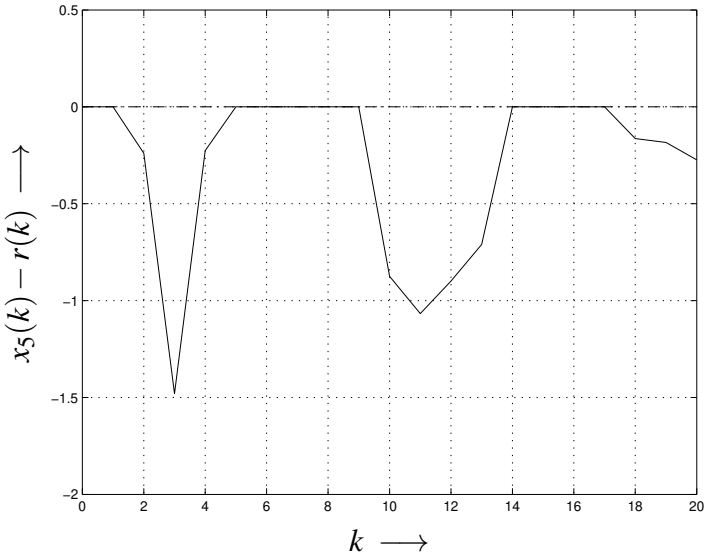


Figure 7. Due date error for a production system with concurrency

In Figure 7 the due date error $x_5(k) - r(k)$ in each cycle $k$ is given for $r(k) = 8 + 6.5k + d(k)$ where $d(k)$ is uniformly distributed zero mean white noise with $|d(k)| \leq 1$. In the cost function $J$ we penalize the due date error for $x_5(k) - r(k) > 0$, which corresponds to the case where a product is ready after its due date. The case where $x_5(k) - r(k) < 0$ is not so bad, as long as there are no problems with a limited output buffer (storage). Keeping that in mind we observe in Figure 7 that the due date error is zero or negative, which means that our product is delivered in time, and therefore the MPC controller is functioning satisfactory.

## 6 Discussion

We have presented a new way to model a class of discrete event systems — the switching max-plus-linear systems — that can operate in different modes, for which in each mode the dynamics can be described by a model that is "linear" in the max-plus algebra. We have discussed three examples of switching max-plus-linear systems, namely a production system, a scheduling system, and a railway network. An MPC design technique has been derived for switching max-plus-linear systems, and we have applied the control design method to the three examples. In general the resulting optimization problem requires a mixed integer optimization algorithm.

The use of switching max-plus-linear systems in modeling and controlling discrete event systems offers some new opportunities. Where in regular max-plus-linear systems one can only model synchronization in a fixed ordering, the use of switching max-plus-linear systems gives the possibility to include concurrency or a structure change. In this way we can account for routing mechanisms (e.g. the switching mechanism in the production system), we can make decisions in scheduling systems (e.g. a railway network) such as changing the order of operations, or breaking synchronizations, and in a scheduling environment we may use the different modes to represent different recipes for production systems.

In future research we will try to find out what conditions are needed on cost criterion $J$, mode function $\Phi$ and mode set $\mathscr{Z}^{(\ell(k))}$ to obtain particular optimization problems (Extended Linear Complementary Problem, Mixed Integer Linear Programming, Mixed Integer Quadratic Programming).

Another topic for future research is the characterization of the class of discrete-event systems that can be modeled by a switching max-plus-linear system, and the expanse and limits of this class. We will study in more detail the properties of switching MPL systems in terms of the underlying MPL subsystems, each with its own max-plus-algebraic eigenvalue, eigenvector, cycle time and communication graph.

*Acknowledgments*

Research partially funded by the Dutch Technology Foundation STW project "Model predictive control for hybrid systems" (DMR.5675), by the European IST project "Modelling, Simulation and Control of Nonsmooth Dynamical Systems (SICONOS)" (IST-2001-37172), by the NWO/STW VIDI project "Multi-agent control of large-scale hybrid systems" (DWV.6188), and by the European 6th Framework Network of Excellence "HYbrid CONtrol: Taming Heterogeneity and Complexity of Networked Embedded Systems (HYCON)" (FP6-IST-511368).

## References


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

Bemporad, A., Morari, M., 1999. Control of integrated logic, dynamics, and constraints. Automatica 35 (3), 407–427.

Chua, L., Deng, A., 1988. Canonical piecewise-linear representation. IEEE Transactions on Circuits and Systems 35 (1), 101–111.

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

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

Davis, L. (Ed.), 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

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

De Schutter, B., van den Boom, T., 2002. Model predictive control for max-min-plus-scaling systems — efficient implementation. In: Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02). Zaragossa, Spain, pp. 343–348.

De Schutter, B., van den Boom, T., 2004. MPC for continuous piecewise-affine systems. System and Control Letters 52 (3/4), 179–192.

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

Gaubert, S., Dec. 1995. Performance evaluation of $(\max, +)$ automata. IEEE Transactions on Automatic Control 40 (12), 2014–2025.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Boston.

Leenaerts, D., van Bokhoven, W., 1998. Piecewise Linear Modeling and Analysis. Kluwer Academic Publishers, Boston.

Maciejowski, J., 2002. Predictive Control with Constraints. Prentice Hall, Pearson Education Limited, Harlow, UK.

van den Boom, T., De Schutter, B., 2002. Properties of MPC for max-plus-linear systems. European Journal of Control 8 (5), 53–462.

van den Boom, T., De Schutter, B., 2004a. Modelling and control of discrete event systems using switching max-plus-linear systems. In: Proceedings of the 7th International Workshop on Discrete Event Systems (WODES'04). Reims, France.

van den Boom, T., De Schutter, B., 2004b. Modelling and control of railway networks. In: Proceedings of the American Control Conference 2004, Boston, MA, USA. pp. 5728–5733.

## Appendix

**Proof of Lemma 1**:
Consider the MMP system

$$x(k) = \mathscr{M}(x(k-1), u(k)) \tag{24}$$

where all entries of the vector function $\mathscr{M}$ are MMP vector functions in terms of the vector $w(k) = \left[x^T(k-1)u^T(k)\right]^T \in \mathbb{R}_{\max}^{n_w}$. Then, similar as for the MMPS systems in De Schutter and van den Boom (2004), we can easily prove that every entry can be written in the min-max canonical form:

$$f_i(w) = \min_{t_i=1,\ldots,K_i} \max_{j=1,\ldots,n_w} (\beta_{i,j,t_i} + w_j)$$

for $i = 1,\ldots,n$, where $\beta_{i,j,t_i} \in \mathbb{R}_{\max}$.

Now we consider vectors $r_\ell$ of the form

$$r_\ell = \left[\, r_{1,\ell}, \ldots, r_{n,\ell} \,\right]^T, \quad \ell = 1,\ldots,n_r$$

where $r_{i,\ell}$ runs form 1 to $K_i$, for $i = 1,\ldots,n$. So we have $n_r = \prod_{i=1}^{n} K_i$ of these vectors. For every $\ell = 1,\ldots,n_r$ we can define a corresponding set $\mathscr{Z}^{(\ell)}$ such that for all $w \in \mathscr{Z}^{(\ell)}$ the minimum of $f_i$ over $t_i$ is reached for $t_i = r_{i,\ell}$. This means that the set $\mathscr{Z}^{(\ell)}$ for $\ell = 1,\ldots,n_r$ is given by

$$\mathscr{Z}^{(\ell)} = \{w \,|\, \max_{j=1,\ldots,n_w} (\beta_{i,j,r_{i,\ell}} + w_j) \leq \max_{j=1,\ldots,n_w} (\beta_{i,j,t_i} + w_j),\ \forall t_i,\ \forall i\}$$

With this definition of $\mathscr{Z}^{(\ell)}$ we then derive that for $w \in \mathscr{Z}^{(\ell)}$ there holds

$$f_i(w) = \max_{j=1,\ldots,n_w} (\beta_{i,j,r_{i,\ell}} + w_j) \tag{25}$$

$$= \left[\, \beta_{i,1,r_{i,\ell}} \ \ldots \ \beta_{i,n_w,r_{i,\ell}} \,\right] \otimes w \tag{26}$$

and for $w \in \mathscr{Z}^{(\ell)}$ we can rewrite

$$f(w) = \Gamma^{(\ell)} \otimes w$$

where the entries of $\Gamma$ are given by $[\Gamma^{(\ell)}]_{i,j} = \beta_{i,j,r_{i,\ell}}$.

By partitioning $\Gamma^{(\ell)} = \left[\, A^{(\ell)} \ B^{(\ell)} \,\right]$ we obtain that for $w \in \mathscr{Z}^{(\ell)}$ we have

$$f(x(k-1), u(k)) = A^{(\ell)} \otimes x(k-1) \oplus B^{(\ell)} \otimes u(k)$$

and the equation (24) can also be written as

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

This proves that the MMP system can be written as a switching MPL system.

□