Event-Driven Model Predictive Control of Timed Hybrid Petri Nets

-draft-

J. Júlvez, S. Di Cairano, A. Bemporad, and C. Mahulea[*]

December 23, 2014

Abstract

Hybrid Petri nets represent a powerful modeling formalism that offers the possibility of integrating in a natural way continuous and discrete dynamics in a single net model. Usual control approaches for hybrid nets can be divided into discrete-time and continuous-time approaches. Continuous-time approaches are usually more precise but can be computationally prohibitive. Discrete-time approaches are less complex but can entail *mode-mismatch* errors due to fixed time discretization. This work proposes an optimization-based *event-driven* control approach that applies on continuous time models and where the control actions change when discrete events occur. Such an approach is computationally feasible for systems of interest in practice and avoids *mode-mismatch* errors. In order to handle modelling errors and exogenous disturbances, the proposed approach is implemented in a closed-loop strategy based on event-driven model predictive control.

Published as:
J. Júlvez, S. Di Cairano, A. Bemporad, and C. Mahulea, "Event-Driven Model Predictive Control of Timed Hybrid Petri Nets," International Journal of Robust and Nonlinear Control, vol. 24, no 12, pp 1724-1742, August 2014. DOI: 10.1002/rnc.2958

[*]J. Julvez and C. Mahulea are with Dpto. Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain. E-mail: {*julvez,cmahulea*}*@unizar.es*.
S. Di Cairano is with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. E-mail: *dicairano@ieee.org*.
A. Bemporad is with IMT Institute for Advanced Studies Lucca, Italy. E-mail: *alberto.bemporad@imtlucca.it*.

1 Introduction

Petri nets represent a widely spread formalism for modeling discrete event systems [28,31]. Similarly to other formalisms for discrete systems, Petri nets suffer from the well-known state explosion problem, i.e., the number of states increases exponentially with respect to the size of the system.

An effective approach to avoid the state explosion problem is to approximate the discrete variables that reach large values by continuous variables. Such variables typically correspond to raw parts, produced items, capacity of buffers, etc. On the other hand, other variables, such as shared resources or processing machines, might maintain small values for any potential system evolution. Hence, they should be kept as discrete. The above considerations lead to hybrid Petri nets [10], a modeling formalism in which the Petri net structure is the same as in a classical Petri net. In hybrid nets, the amount of tokens in the subset of *continuous places* and the firings of the subset of *continuous transitions* are real numbers, while the amount of tokens in the subset of *discrete places* and the firings of the subset of *discrete transitions* are integer numbers as in classical Petri nets.

As in timed discrete Petri nets, several semantics can be associated to the firing of continuous transitions in timed hybrid Petri nets. In this paper, we will consider finite-server semantics. Under this semantics the firing rate of a continuous transition remains constant as long as no place gets empty [2]. When a place gets empty, the firing rate changes and remains constant again until another place gets empty. In this way, the continuous-time evolution of the marking of a continuous Petri net is piecewise-linear.

The autonomous behavior of a hybrid Petri net can be modified by introducing control actions on the net transitions. By using a common interpretation where a continuous transition is seen as a valve through which a liquid flows, the control action on the transition determines how much such valve is open. On the other hand, control actions on discrete transitions can delay their firing instant. The introduction of control actions allows one to define control problems in the framework of hybrid Petri nets.

In this paper we propose a framework for optimization-based control of timed hybrid Petri nets, based on their piecewise linear trajectories. The task of solving control problems for continuous-time piecewise-linear systems is, in general, a challenging problem [21,32]. A common approach to overcome this difficulty is to consider a discrete-time representation of the system [27]. However, time discretization leads to mode mismatch errors [12], mode changes that occur during the intersampling, and hence are lost or delayed in the discrete-time representation, leading to possibly large differences between the discrete-time and continuous-time trajectories. Clearly, the smaller the time step, the smaller the effects of the mode mismatch are. Unfortunately, in the case of finite horizon optimal control, reducing the sampling period usually increases the complexity of the problem to solve [5,12].

The framework proposed in this paper is event-driven, and hence, by considering mode switches as included in the events, mode-mismatch is avoided. In such an approach, the control input is parametrized by a piecewise constant function where the time-duration of the different step is not assumed constant. As a consequence, the control signal is given by tuples $(v(k), q(k), \sigma(k))$ defining the integral of the control signal during the application period for continuous transitions, the application period, and the discrete transitions to be fired at the beginning of period, respectively. A tuple is produced when an event occurs, i.e., for a hybrid Petri net, when a place gets empty, when a discrete transition fires or when a discrete transition becomes enabled or disabled. Given that the marking evolution is piecewise-linear, the full system trajectory is defined by the sequence of tuples. Preliminary results of this method for optimal control of continuous Petri nets were proposed in [22]. This paper improves such preliminary results by considering hybrid Petri nets, and further extending the optimal control framework to a model predictive one, in order to provide a closed-loop strategy that corrects external disturbances.

Model predictive control (MPC) [8, 24] is an optimization based receding horizon closed-loop control strategy, where at each control cycle a (constrained) finite horizon optimal control problem is solved, and only the first part of the computed optimal input profile is applied to the system. However, differently from open-loop optimal control, when fresh information on the system state becomes available, by measurements or estimatoion, the optimal sequence is recomputed. In this way, feedback is taken into account and MPC results to be a closed-loop control strategy.

Due to the improved performance achieved by using optimization algorithms, to the capability of handling multiple inputs, and to the possibility of enforcing constraints, model predictive control has found several applications, for instance in process industry [29], automotive (e.g., [14, 15]), aerospace (e.g., [19]), and supply chains (e.g., [7]). A previous application of model predictive control to a particular class of discrete-event systems is found in [11].

Several approaches to control continuous Petri nets exist in the literature. In [33] an algorithm to track control of Petri nets without joins is suggested. The work in [2] develops a method based on a linear programming problem to obtain optimal modes of operation for hybrid Petri nets and also proposes efficient techniques for sensitivity analysis on this kind of nets. Classical discrete-time model predictive control has been applied to continuous Petri nets under infinite server semantics in [16,25]. Here, the focus is on finite server semantics for which classical model predictive control requires time discretization that may lead to mode mismatch errors when places of the net become empty during the intersampling. As such, the sampling period must be kept small enough to minimize the problems due to such errors, which however increases the computational complexity of the MPC controller. With respect to previous approaches, the present paper provides improvements in the class of models considered -hybrid Petri nets instead of continuous Petri nets- and on the controller properties, since the event-driven MPC does not require time-discretization nor oversampling to avoid mode mismatch problems. In particular, our approach extends the work in [2] by embedding both continuous and discrete transitions in the same set of equations using an event-driven formulation, and by proposing a MPC framework based on this formulation. The use of MPC framework allows for closed-loop control, and hence disturbance rejection, by repeated optimization in a receding horizon strategy [4].

The rest of the paper is organized as follows. Section 2 introduces hybrid Petri nets. A technique to express the behavior of hybrid Petri nets in an event-driven fashion is discussed in Section 3. Section 4 presents two methods for the event-driven control of continuous Petri nets. A finite horizon open-loop optimal control problem is introduced first, then used to implement a model predictive control strategy. Two case studies are shown in section 5. The conclusions are summarized in section 6.

Notation: $\mathbb{R}$, ($\mathbb{R}_{0+}$, $\mathbb{R}_{+}$) is the set of (nonnegative, positive) real numbers and $\mathbb{N}$ is the set of natural numbers. For a set $\mathcal{S}$, $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. Inequalities between vectors are intended componentwise and when a number c is used in the place of a vector, it indicates a vector where all the components have value c. The transpose of a matrix A is denoted as A'. For a time-dependent vector x, $x[i](k)$ denotes the value of component i at step k, and $x(k)$ denotes the whole vector at step k. The step (k) will be omitted if clear from the context. For a vector $\mu \in \mathbb{R}^n$, $\mu(h|\tau)$ is the h-steps ahead predicted value starting from time τ. Since this paper discusses event-driven control, the steps start at the occurrence of events and time duration of the steps is not constant.

2 Hybrid Petri nets

This section introduces the basic concepts related to *hybrid* Petri nets. In the following we assume the reader is familiar with the basic concepts of Petri nets (PNs), see [28, 31] for an extensive overview.

2.1 Untimed hybrid Petri nets

In contrast to conventional (i.e., discrete) PNs, the arc weights of hybrid PNs are real-valued.

Definition 1 (HPN) *A* Hybrid Petri Net *(HPN) is a tuple* $\mathcal{N} = \langle P, T, Pre, Post\rangle$ *where:*

- P *is a set of* $|P|$ *places, and* T *is a set of* $|T|$ *transitions.*
- $Pre : P\times T \to \mathbb{R}_{0+}$ *and* $Post : P \times T \to \mathbb{R}_{0+}$ *are the* pre- *and* post- *incidence functions that specify the arc weights.*
- $P = P_c \cup P_d$, $P_c \cap P_d = \emptyset$, *and* $T = T_c \cup T_d$, $T_c \cap T_d = \emptyset$.

The set of places P is partitioned into a set of *discrete places*, P_d, and a set of *continuous places*, P_c. Similarly, the set of transitions T is partitioned into a set of *discrete transitions*, T_d, and a set of *continuous transitions*, T_c. Discrete places are graphically represented as circles and continuous places as double circles, and similarly discrete transitions are represented as rectangles and continuous transitions as double rectangles, see for instance the network in Figure 3.

The main difference between HPNs and discrete PNs is in the way the transitions are fired. In discrete PNs the transitions are fired a natural number of times. In HPNs the discrete transitions are also fired a natural number of times, but the continuous transitions can be fired a real number of times which leads to real markings in continuous places.

In order to ensure the integrality of the marking of discrete places, two conditions are required:

a) $Pre[p, t] \in \mathbb{N}$ and $Post[p, t] \in \mathbb{N}$ for every $p \in P_d$ and every $t \in T_d$;

b) $Pre[p,t] = Post[p,t]$ for every $p \in P_d$ and every $t \in T_c$.

The incidence matrix of the net is $\mathbb{C} = Post - Pre$, $\mathbb{C} \in \mathbb{R}^{|P| \times |T|}$ and the state of the net is the marking $m \in {\mathbb{R}_{0+}}^{|P_c|} \times \mathbb{N}^{|P_d|}$, which evolves dynamically. The marking can be partitioned into its real and natural components, $m = [m_c' \; m_d']'$, $m_c \in {\mathbb{R}_{0+}}^{|P_c|}$, $m_d \in \mathbb{N}^{|P_d|}$, the marking of continuous places and discrete places, respectively. The *preset* and *postset* of a node $\xi \in P \cup T$ are denoted as ${}^\bullet\xi$ and $\xi^\bullet$.

Definition 2 (HPN system) *A* Hybrid Petri Net System *is a tuple $\langle \mathcal{N}, m_0 \rangle$ where:*

- *$\mathcal{N}$ is a HPN.*
- *$m_0 : P \to \mathbb{R}_{0+}$ assigns to each place p, an initial marking $m_0[p]$. For every $p \in P_d$, it is required that $m_0[p] \in \mathbb{N}$.*

Definition 3 (Enabling degree) *Let $\langle \mathcal{N}, m_0 \rangle$ be a HPN system. At marking m, the enabling degree of a transition $t \in T_d$ is $enab(t,m) = \min_{p \in {}^\bullet t} \left\lfloor \frac{m[p]}{Pre[p,t]} \right\rfloor$, and the enabling degree of a transition $t \in T_c$ is $enab(t,m) = \min_{p \in {}^\bullet t} \frac{m[p]}{Pre[p,t]}$.*

Definition 4 (Firing) *Let $\langle \mathcal{N}, m_0 \rangle$ be a HPN system. A transition $t \in T$ can be fired in any amount α such that $0 \leq \alpha \leq \mathrm{enab}(t,m)$, where $\alpha \in \mathbb{N}$ if $t \in T_d$, $\alpha \in \mathbb{R}$ if $t \in T_c$. The firing of t in a certain amount α leads to a new marking $m' = m + \alpha \cdot \mathbb{C}[P,t]$, where $\mathbb{C}[P,t]$ is the column of the incidence matrix corresponding to transition t.*

Hence, as in discrete PN systems, the state (or fundamental) equation $m = m_0 + \mathbb{C} \cdot \sigma$ summarizes the marking evolution where σ is the firing count vector. Similarly to continuous Petri nets, in HPNs the marking of a continuous place can be seen as an amount of fluid being stored, and the firing of a continuous transition can be considered as a flow of this fluid going from a set of places (input places) to another set of places (output places).

As in classical PNs, vectors $Y \geq 0$, $Y \cdot \mathbb{C} = 0$ ($X \geq 0$, $\mathbb{C} \cdot X = 0$) represent P-semiflows or conservative components (T-semiflows or consistent components). A net $\mathcal{N}$ is conservative (consistent) if there exists $Y > 0$ such that $Y \cdot \mathbb{C} = 0$ ($X > 0$ such that $\mathbb{C} \cdot X = 0$). A net $\mathcal{N}$ is structurally bounded if there exists $Y > 0$ such that $Y \cdot \mathbb{C} \leq 0$ (notice that this condition can be checked in polynomial time).

2.2 Timed hybrid Petri nets

For the timing interpretation of continuous transitions a first order (or deterministic) approximation of the discrete case [30] is used, hence assuming that the delays associated to the firing of the transitions are approximated by their mean values. As a result, the marking evolution with respect to time τ is

$$m(\tau) = m(0) + \mathbb{C} \cdot \sigma(\tau). \tag{1}$$

where $\sigma(\tau)$ is the firing count vector at time τ. The instantaneous flow $f \in \mathbb{R}_{0+}$ of a continuous transition $t \in T_c$ is defined as the derivative of its firing count vector with respect to time, i.e., $f = \dot{\sigma}$.

Different semantics have been defined for the firing of continuous transitions, the most commonly used being *infinite server* (also called variable speed) [30] and *finite server* (also called constant speed) [10] semantics. In this paper, finite server semantics is considered. Under finite server semantics, every continuous transition, $t \in T_c$, of the timed system is associated with a real parameter $\lambda[t] > 0$ that is the maximum flow allowed by t, i.e., $f[t] \leq \lambda[t]$.

As for continuous transitions, different time interpretations can be adopted for the firing of discrete transitions. Here, single server semantics for discrete transitions is considered and a deterministic delay $\vartheta[t] \in \mathbb{R}_+$ is associated to each transition $t \in T_d$. An enabled discrete transition t can fire if it has been enabled for at least $\vartheta[t]$ time units. No resolution policy for the conflicting transitions is specified letting the exact firing time to be determined by the controller which aims to optimize a given objective function. Notice that the firing of a discrete transition might disable other transitions in conflict.

Definition 5 (THPN system) *A* Timed Hybrid Petri Net System *(THPN system) is a tuple $\langle \mathcal{N}, m_0, \lambda, \vartheta \rangle$ where:*

- *$\langle \mathcal{N}, m_0 \rangle$ is a HPN system.*
- *$\lambda : T_c \to \mathbb{R}_+$ defines the maximum flow allowed by each continuous transition.*
- *$\vartheta : T_d \to \mathbb{R}_+$ defines the time delay of each discrete transition.*

Intuitively, if a continuous transition is seen as a valve through which a fluid passes, λ can be seen as the maximum flow admitted by the valve. In contrast to [2], we do not impose a lower bound for the flow of the transitions, thus, $0 \leq f[t] \leq \lambda[t]$.

In THPN two types of enabling for continuous transitions are considered [2].

Definition 6 (Enabling of continuous transitions) *Let $\langle \mathcal{N}, m_0, \lambda \rangle$ be a THPN system and $t \in T_c$. Let m be a marking such that $m[p] \geq Pre[p,t]$ for every $p \in {}^\bullet t \cap P_d$.*

- *t is* strongly enabled *at m if $m[p] > 0$ for every $p \in {}^\bullet t \cap P_c$.*
- *t is* weakly enabled *at m if there exists $p \in {}^\bullet t \cap P_c$ such that $m[p] = 0$.*

A continuous transition is not enabled if there exists $p \in {}^\bullet t \cap P_d$ such that $m[p] < Pre[p,t]$. Notice that in contrast to an untimed HPN, in a THPN a continuous transition having an empty input continuous place may be weakly enabled and can fire. This happens when such an input place receives some input flow that is instantaneously consumed by the transition.

The flow of a transition depends on its enabling state.

Definition 7 (Flow) *Let $\langle \mathcal{N}, m_0, \lambda \rangle$ be a THPN system and $t \in T_c$, then:*

- *If t is strongly enabled then it has maximum flow, i.e., $f[t] = \lambda[t]$.*
- *If t is not enabled then it has no flow, i.e., $f[t] = 0$.*

- *The flow of the weakly enabled transitions must ensure that $m[p] \geq 0$, for all $p \in P_c$.*

The computation of an admissible flow f is non-trivial when several empty places appear. In [1], an iterative algorithm is suggested to compute one admissible flow f. In this paper, f is computed similarly to [2] where the set of admissible flows is characterized by a set of linear inequalities. Similarly to the firing of discrete transition, the flow of continuous transitions will be determined by the controller.

Let us consider two of the events that can happen during the evolution of a THPN: a) A discrete transition is fired; b) A continuous place becomes empty. Between two consecutive of such events no discrete transition is fired and no continuous place becomes empty. Hence, according to Definition 7, between such events the flow of continuous transitions f keeps constant and consequently the trajectory of the marking of the continuous places is linear. The occurrence of an event can modify the value of f which will keep constant until a new event occurs. This way, the overall trajectory of the marking of the continuous places is piecewise linear.

Example 1 *Consider the system in Figure 1. The only input place of t_1 is marked, hence t_1 is strongly enabled and $f[t_1] = \lambda[t_1] = 2$. Given that t_2 is always strongly enabled, the evolution of $m[p_1]$ is given by $\dot{m} = \lambda[t_2] - \lambda[t_1] = -1$. At time 1, p_1 becomes empty, i.e., an event occurs, and t_1 becomes weakly enabled. Now, the maximum flow admitted by t_1 is 1, since a greater flow would cause $m[p_1]$ to be negative. Being $f[t_1] = 1$, p_1 remains empty. Now, p_1 can be seen as a tube instead of a deposit and no more events occur. For arbitrary values of $\lambda[t_1]$ and $\lambda[t_2]$, when p_1 is empty the flow of t_1 is defined as $f[t_1] = \min(\lambda[t_1], \lambda[t_2])$.*

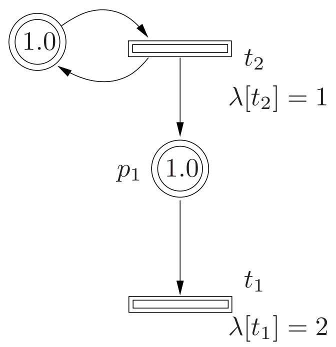


Figure 1: Transition t_1 becomes weakly enabled at $\tau = 1$.

2.3 Control actions

Control actions can be introduced in THPNs in order to modify the autonomous evolution. In THPNs the control affects the transitions.

A discrete transition $t \in T_d$ is *controllable* when its firing time is a decision variable that can be selected by an appropriate control strategy. In order to ensure that the task modeled by the transition is finished, the firing is not allow to happen before $\vartheta[t]$ time units have elapsed from the enabling of t.

A continuous transition $t \in T_c$ is *controllable* when its flow f is a decision variable $u[t]$ such that

$$0 \leq u[t] \leq \lambda[t], \tag{2}$$

where $u \in \mathbb{R}^m$ is the vector of controls. An action $u[t]$ on the transition t can be seen as if the hypothetical valve associated to t was opened by the amount $u[t]$. In this paper it is assumed that all transitions (discrete and continuous) are controllable. If t is strongly enabled, (2) is the only constraint that $u[t]$ must satisfy. However, if t is weakly enabled, $u[t]$ must be such that the nonnegativity of the marking is ensured.

Example 2 *In order to show how input actions modify the evolution of a system, we apply the input actions $u[t_1] = 1.5$ and $u[t_2] = 1$ to the system in Figure 1. After two time units p_1 becomes empty. Hence, the maximum flow allowed by t_1 is the input flow coming to p_1, that is 1, i.e., $u[t_1]$ must satisfy $0 \leq u[t_1] \leq 1$. Let $u[t_1] = 0.5$, and consequently p_1 will start to fill at a rate of 0.5 tokens per time unit.*

3 Event-driven representation

This section describes how THPNs can be expressed as a particular class of Mixed Logical Dynamical systems where each step represents the marking evolution between two events of the THPN. Section 3.1 introduces the event-driven Mixed Logical Dynamical (eMLD) systems, and Section 3.2 shows how the THPN is transformed into an eMLD.

3.1 Event-driven mixed logical dynamical systems

Mixed Logical Dynamical (MLD) systems [6] are computationally oriented representations of discrete-time hybrid systems. MLDs consist of a set of linear equalities and inequalities involving both real and Boolean ($\{0, 1\}$) variables. An MLD system is described by the relations

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_4 \tag{3a}$$
$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_4 \tag{3b}$$
$$E_1u(k) + E_5x(k) \leq E_2\delta(k) + E_3z(k) + E_4, \tag{3c}$$

where $x = [x_c' \; x_d']' \in \mathbb{R}^{n_r} \times \{0, 1\}^{n_b}$ is a vector of continuous and binary states, $u = [u_c' \; u_d']' \in \mathbb{R}^{m_r} \times \{0, 1\}^{m_b}$ are the inputs, $y = [y_c' \; y_d']' \in \mathbb{R}^{p_r} \times \{0, 1\}^{p_b}$ are the outputs, $\delta \in \{0, 1\}^{r_b}$, $z \in \mathbb{R}^{r_r}$ represent auxiliary binary and continuous variables, respectively, and A, C, B_i, D_i, $i = 1, \dots, 4$, E_i, $i = 1, \dots, 5$ are matrices of suitable dimensions. Given the current state $x(k)$ and input $u(k)$, the evolution of (3a)-(3c) is determined by solving (3c) for $\delta(k)$ and $z(k)$, then updating $x(k+1)$ and $y(k)$ from (3a) and (3b). It is assumed that the system (3a)-(3c) is *well-posed* [6], which means that for any value of $x(k)$, $u(k)$ within the range of interest, $\delta(k)$, $z(k)$ are uniquely determined by (3c).

In [13] the authors have proposed an event-driven MLD model (eMLD),

$$\chi(k+1) = \chi(k) + B_1\mu(k) + B_2\delta(k) + B_3z(k) + B_4 \tag{4a}$$
$$y(k) = C\chi(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_4 \tag{4b}$$
$$E_1\mu(k) + E_5x(k) \leq E_2\delta(k) + E_3z(k) + E_4, \tag{4c}$$

where $\chi(k) = [x(k)' \ \tau(k)]'$, $q \in \mathbb{R}_{0+}$, $\mu(k) = [v(k)' \ u_d(k)' \ q(k)]'$. In the eMLD (4), the counter k represents the number of events, the additional state variable $\tau(k)$ is the total time elapsed when the k^{th} event occurs, $q(k)$ is the time between the k^{th} and the $(k+1)^{th}$ events, and $v(k)$ is the integral of the continuous control input between the k^{th} and the $(k+1)^{th}$ events, where it is assumed piecewise constant, i.e., $v(k) = q(k)u_c(t(k)^+)$. As discussed in [13], in the eMLD system an event occurs either when the value of δ changes, due to (4c), or when the input μ is changed.

In what follows we show how to transform THPNs in eMLD form.

3.2 Transforming THPNs to event-driven MLDs

3.2.1 Statements as linear inequalities

Consider the THPN in Figure 1. According to the defined semantics, the continuous-time marking evolution is described by

$$\begin{aligned} \text{if} \quad m[p_1] > 0 \quad \text{then} \quad & \dot{m}[p_1] = -1 \\ \text{else} \quad & \dot{m}[p_1] = 0 \end{aligned} \tag{5}$$

Clearly, if the initial marking of p_1 is $m_0[p_1] = 1$, after 1 time unit p_1 gets empty. Such an evolution can be described appropriately by a discrete-time model only if the duration of the sampling period $h \in \mathbb{R}$ satisfies $h \cdot k = 1$, for some $k \in \mathbb{N}$. If this is not the case, the marking of p_1 will become at some point negative, and the evolution will block. This phenomenon is named mode-mismatch error, where the exact instant of the mode switch is lost, because it occurs in the intersampling. Mode-mismatch is present in most discrete-time models of hybrid systems [5], and it can be alleviated by imposing a very small sampling period h (oversampling), which however results in unnecessary computations in the control algorithm.

Indeed, mode mismatch is not present if a continuous time model of the system is used. However, for continuous time models, the input is an infinite-dimensional decision variable, hence computational tools which require finite dimensional decision variables, such as mathematical programming, cannot be used. To overcome the mode-mismatch while retaining finite dimensionality of the input, an event-driven approach, instead of a discrete-time one, shall be used. In an event-driven approach the system evolves at events, and the time separation of the events is not constant, but modelled as a variable, e.g., component $q(k)$ of $\mu(k)$ in (4). By including mode switches in the set of events, the mode-mismatch error is removed.

For the marking evolution of the system in Figure 1 we obtain

$$\begin{aligned} \text{if} \quad m[p_1](k) > 0 \quad \text{then} \quad & m[p_1](k+1) = m[p_1](k) - 1 \cdot q(k) \\ \text{else} \quad & m[p_1](k+1) = 0. \end{aligned} \tag{6}$$

Such a conditional statement can be easily included in an eMLD system (4). To construct the eMLD representation of a THPN, we write the evolution of the system between events k and $k+1$. Recall that the time between these two events (denoted by $q(k)$) is not constant. The possible events at $k+1$ are:

1. A marked continuous place $p \in P_c$ at k becomes empty at $k+1$. In this case, it is necessary to recompute the firing flow of continuous transitions to ensure the positiveness of $m[p]$;
2. A discrete transition $t \in T_d$ that has been enabled during a time period greater than or equal to $\vartheta[t]$ is fired;
3. A discrete transition changes its enabling status, i.e., it becomes enabled or disabled. In this case, its associated clock should be started or disabled.

After $q(k)$ time units, at least one of the previous events must happen. Now, we define the set of constraints of the eMLD representation of a THPN. The first constraint is the state equation corresponding to the continuous places

$$m_c(k+1) = m_c(k) + \mathbb{C} \cdot v(k) \geq 0. \tag{7}$$

Notice that we used a variable $v(k) = q(k) \cdot u(k)$ in order to linearize the state equation. Knowing $q(k)$ and $v(k)$, the control action of continuous transitions is immediately obtained. The constraints on the control action (2) are also translated to the new parametrization of the control actions by including

$$0 \leq v(k) \leq q(k) \cdot \lambda, \tag{8}$$

in the set of constraints. The constraint that all markings must be nonnegative is included in (7). Obviously, the flow of discrete transition is null and this can be done adding the following constraints

$$v_j(k) = 0, \forall t_j \in T_d. \tag{9}$$

Finally, if a continuous transition t_j has a discrete input place p_i and $m_i(k)$ is lower than the weight of the arc (p_i, t_j), the flow of t_j must be null.

$$\text{if} \quad m_i(k) < Pre[p_i, t_j] \quad \text{then} \quad v_j(k) = 0 \quad (\forall t_j \in T_c, \text{ and } \forall p_i \in {}^\bullet t_j, p_i \in P_d). \tag{10}$$

Next we model the three possible events of the THPN.

3.2.2 A continuous place gets empty

In order to identify the fact that a continuous place $p_i \in P_c$ that was marked at k and gets empty at $k+1$, we use a boolean variable β_i defined as

$$m_i(k) > 0 \quad \text{and} \quad m_i(k+1) = 0 \quad \Longleftrightarrow \quad \beta_i(k) = 1. \tag{11}$$

Later, we will make use of this boolean variables to force the occurrence of at least one event at the end of each time period.

3.2.3 Firing of a discrete transition

To manage the firing of discrete transitions we will use a vector $d(k) \in \mathbb{R}_{\geq 0}^{|T^d|}$ to keep track of the time elapsed from the enabling of discrete transitions. Obviously, $d(0) = 0$. If a transition $t_i \in T_d$ becomes enabled at τ, it cannot fire before $\tau + \vartheta[t_i]$. We define a boolean variable γ_i such that $\gamma_i(k) = 0$ if $d_i(k) < \vartheta[t_i]$

$$\text{if} \quad d_i(k) < \vartheta_i \quad \text{then} \quad \gamma[t_i](k) = 0. \tag{12}$$

Therefore, if $\gamma_i = 0$, then $t_i \in T_d$ will not fire. On the other hand, if $d_i(k) \geq \vartheta[t_i]$ we do not assign any value to γ_i, being a decision variable. In this last case, if γ_i is assigned to 1, the corresponding discrete transition will fire. The firing of discrete transitions is described by

$$m(k+1) = m(k) + \mathbb{C} \cdot \gamma(k) \geq 0, \tag{13}$$

where $\gamma(k)$ is the vector having as elements the values of $\gamma_i(k)$ for discrete transition and zero components for continuous ones. Since the firing of a discrete transition should occur instantaneously, i.e., this firing does not consume time, we introduce the constraint

$$\text{if} \quad \gamma' \cdot \mathbf{1} \geq 1 \quad \text{then} \quad q(k) = 0. \tag{14}$$

3.2.4 Updating the clocks of discrete transitions

In order to capture the enabling status of a discrete transition t_i, let us define $\alpha_i(k)$ such that $\alpha_i(k) = 1$ if $t_i(k) \in T_d$ is enabled and $\alpha_i(k) = 0$ otherwise. This is can be easily achieved by

$$m(k) \geq Pre[\cdot, t_i] \quad \Longleftrightarrow \quad \alpha_i(k) = 1. \tag{15}$$

If a transition is enabled during the time interval from k to $k+1$ then its clock is increased by $q(k)$. Otherwise, it is set to zero. This is modeled by

$$\begin{aligned} \text{if} \quad \alpha_i(k) = 1 \quad \text{and} \quad \alpha_i(k+1) = 1 \quad & \text{then} \quad d_i(k+1) = d_i(k) + q(k) \\ & \text{else} \quad d_i(k+1) = 0. \end{aligned} \tag{16}$$

Let us define a boolean variable μ_i such that $\mu_i(k) = 1$ if $t_i(k) \in T_d$ changes its enabling status from k to $k+1$

$$\begin{aligned} \big((\alpha_i(k) = 0 \quad \text{and} \quad \alpha_i(k+1) = 1) \text{ or } (\alpha_i(k) = 1 \quad \text{and} \quad \alpha_i(k+1) = 0\big) \\ \Longleftrightarrow \mu_i(k) = 1. \end{aligned} \tag{17}$$

Consequently, in order to ensure that at least one of these three events occurs at k the following constraint is introduced

$$\sum_i \beta_i(k) + \sum_i \gamma_i(k) + \sum_i \mu_i(k) \geq 1. \tag{18}$$

4 Event-driven control of THPNs

In this section, we show how optimization-based control can be applied to THPNs via their eMLDs formulation, and how feedback can be accounted for by a model predictive control strategy. We first propose a set of cost functions and constraints that can be used to formulate open-loop finite horizon optimal control problems for the THPN, which can be solved by standard Mixed Integer Linear Programming (MILP) algorithms. Similarly to formulations of optimal control problems of discrete-time systems that explicitly specify the final time instant of the period over which the optimization is carried out is, we will explicitly specify the number of events, N, over which the optimization is performed. Notice that in the proposed event-driven framework, the actual time period elapsed till the $i-th$ event takes place depends on the duration of each time interval which is not constant. Given that such durations are variables of the optimization problem, many different constraints can be set on them. Moreover, as it is shown next, the event-driven approach allows to formulate minimum-time control problems. At the end of the section, we show how the optimal control problem can be used as the base for a receding horizon control strategy, hence implementing an event-driven model predictive control algorithm for the THPN.

4.1 Event-driven optimal control

The advantages of formulating the THPN as an eMLD system (4) is that the dynamics are expressed by mixed-integer equalities and inequalities. As a consequence, the dynamics equations can be included into the mixed integer optimization problem

$$\begin{aligned}
\min_{\bar{\mu}(t)} \quad & J(\bar{\mu}(t), \bar{\chi}(t)) && \text{(19a)}\\
\text{s.t.} \quad & \chi(k+1|t) = A\chi(k|\tau) + B_1 v(k) + B_2\delta(k|\tau) + B_3 z(k|\tau) + B_5 && \text{(19b)}\\
& E_2\delta(k|\tau) + E_3 z(k|\tau) \leq E_1\mu(k|\tau) + E_4\chi(k|\tau) + E_5 && \text{(19c)}\\
& H_2\delta(k|\tau) + H_3 z(k|\tau) \leq H_1\mu(k|\tau) + H_4\chi(k|\tau) + H_5 && \text{(19d)}\\
& \chi(0|t) = \chi(t), \quad k = 1, \ldots, N && \text{(19e)}
\end{aligned}$$

where J in (19a) is the cost function, $\bar{\mu}(t) = \{\mu(k|\tau)\}_{k=0}^{N}$ are the decision variables, $\bar{\chi} = \{\bar{\chi}(k|\tau)\}_{k=0}^{N}$ is the eMLD state trajectory, (19b) and (19c) define the dynamics, (19d) models additional constraints, and (19e) defines the initial state used for prediction. Note that once the initial state is fixed by (19e) the trajectory $\bar{\chi}(t)$ and the value of the auxiliary variables is assigned because of the wellposedness of the eMLD.

The decision vector $\bar{\mu}(t)$ contains the continuous command integral, the corresponding duration, and the discrete command. From the first ones and the second one, the flow commands can be easily obtained as $u(h|t) = v(h|t)/q(h|t)$, with application interval $(\tau(h|t), \tau(h|t) + q(h|t))$.

Since the constraints in (19) are linear with integer and real variables, by choosing the cost function (19a) to be linear, the resulting problem is a mixed integer linear programming (MILP) problem. Thus, the complexity of solving a MILP is exponential in the number of boolean variables. However, efficient algorithms and software exists (see references [9, 20, 23, 26]) that allow the application of MILP even to large scale real industrial systems modeled by Petri nets [18]. Modern MILP software are capable of solving problems with thousands of variables in few tents or hundreds of seconds. Thus, although not suitable for systems with fast dynamics (millisecond range), MILP seems suitable for systems with dynamics in the medium to slow range (seconds, minutes). In particular, in the proposed approach, at each step, there exists (i) one boolean variable per continuous place (variables β in (11) to identify that a continuous place gets empty), (ii) three boolean variables per discrete transition (variables γ in (12) that determines if the transition fires; variables α in (15) to capture the enabling status; and variables μ in (17) showing that the enabling status has changed). Therefore, the number of boolean variables is $N \cdot (|P_c| + 3 \cdot |T_d|)$. Notice that in order to store the marking of discrete places it is not necessary to restrict the variables to integer values in the MILP because integer values will be automatically forced by the integer firings. Finally, the computational complexity can be reduced by introducing specific cuts in the optimization problem, which may reduce the overall performance in favor of faster calculation.

Figure 2 sketches the steps that have been followed to obtain an MILP problem from the initial THPN. If (19a) is chosen to be a quadratic function, the resulting problem will be a mixed-integer quadratic problem, which is still solvable, even though computationally more complex [13].

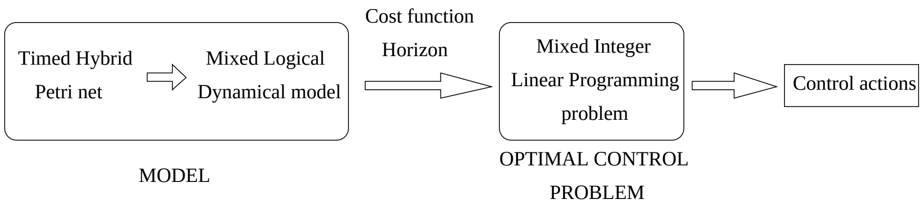


Figure 2: Obtaining an MILP problem from a THPN.

The cost function (19a) and the additional constraints (19d) are used to define the objectives of the optimization problem, as shown next.

4.1.1 Final target marking

In order to enforce the marking to reach a desired target marking $\widetilde{m}$ after N events, a terminal constraint can be added

$$m(N) = \widetilde{m}. \tag{20}$$

The terminal constraint (20) can be softened to preserve feasibility of optimization problem (19), hence adding to J in (19a) the term

$$F\Big(m(N), \tau(N)\Big) = \rho \, \|m(N) - \widetilde{m}\|_\infty, \tag{21}$$

where ρ is a large weight. A more general case is to consider a desired marking range, for instance a polyhedral set expressed by the constraints $\mathcal{M}_N m(n) \leq M_N$, $\mathcal{M}_N \in \mathbb{R}^{q\times|P|}$, $M_N \in \mathbb{R}^q$.

4.1.2 Cost function

For THPN control, the general form of the cost function is

$$J(\bar{\chi}(t), \bar{\mu}(t)) = F\Big(m(N|t), \tau(N|t)\Big) + \sum_{k=0}^{N-1} L\Big(m(k|\tau), \tau(k|\tau), \mu(k|\tau)\Big). \quad (22)$$

hence composed of a terminal cost F and a stage cost L, usually in the form

$$\begin{aligned} L(m, \tau, \mu) &\triangleq \|m - \widetilde{m}\|_p^{Q_1} + \|\tau - \widetilde{\tau}\|_p^{Q_2} + \|v - \widetilde{v}\|_p^{R_1} + \|q - \widetilde{q}\|_p^{R_2} && (23a) \\ F(m, \tau) &\triangleq \|m - \widetilde{m}\|_p^{Q_N} + \|\tau - \widetilde{\tau}\|_p^{Q_\tau},\ p \in \{1, \infty\}. && (23b) \end{aligned}$$

where "$\widetilde{\ }$" denotes a given reference for the corresponding vector. The following subsections show some of the possible control goals in an event-driven framework. The use of $1, \infty$-norms allows to formulate (22) as a linear function, through auxiliary variables and linear constraints [3].

A case of particular interest is *minimum-time* control, where the minimum time to reach a certain marking is sought. Thus, together with terminal constraint (20) the stage cost and terminal cost are respectively set to

$$L(m(k), \tau(k), \mu(k)) = q(k),\ F\Big(m(N), \tau(N)\Big) = 0. \quad (24)$$

A different criterion to reach the desired marking $\widetilde{m}$ is *minimum-effort*, which minimizes the intensity of the command input $u(\tau)$, hence letting the THPNs evolve as close as possible to its autonomous behavior. By using the ℓ_1-norm of the input signal, we obtain $J(m, \tau, q, v) = \int_0^{\tau_N} \|u(\tau)\| \mathrm{d}t = \sum_{k=0}^{N-1} \int_{\tau(k)}^{\tau(k+1)} \|u(\tau)\|_1 \mathrm{d}\tau$. Since u is constant in each period $[\tau(k), \tau(k+1))$,

$$L\Big(m(k|\tau), \tau(k|\tau), \mu(k|\tau)\Big) = \|v(k|\tau)\|_1,\ F\Big(m(N|t), \tau(N|t)\Big) = 0. \quad (25)$$

A slightly different cost function from (22) can be used to represent the *minimum-displacement* criterion. This criterion looks for the trajectory that minimizes the largest deviation from a desired continuous state trajectory $\widetilde{m}(\cdot)$, that we assume piecewise linear and continuous (a special case is $\widetilde{m}(\cdot) \equiv \widetilde{m}$)

$$J(\bar{\chi}(t), \bar{\mu}(t)) = \max_{\tau \in [\tau(0), \tau(N)]} \|m(\tau) - \widetilde{m}(\tau)\|_\infty. \quad (26)$$

Proposition 1 *Let $m_c(\tau)$, $\forall \tau \in [\tau_0, \tau_N]$, be the trajectory of continuous states of a THPN system, $\tau_0 < \tau_1 < \ldots < \tau_N$ be the event instants, assume that $\widetilde{m}(\tau)$ is linear over each $[\tau_i, \tau_{i+1})$, $i = 0, \ldots, N-1$ and continuous over $[\tau_0, \tau_N]$. Then*

$$\max_{\tau \in [\tau_0, \tau_N]} \|m(\tau) - \widetilde{m}(\tau)\|_\infty = \max_{k=0,\ldots,\tau_N} \{\|m(k|\tau) - \widetilde{m}(k|\tau)\|_\infty\}. \quad (27)$$

Proof: The marking trajectories of a THPN are continuous, so $\|m(\cdot) - \widetilde{m}(\cdot)\|_\infty$ is continuous, being the composition of continuous functions ($\|\cdot\|_\infty$, $m(\cdot)$, $\widetilde{m}(\cdot)$), and therefore the maximum over $[t_0, t_N]$ is well defined. Moreover, function $\|m(\cdot) - \widetilde{m}(\cdot)\|_\infty$ is a convex function of time τ on $[\tau(k), \tau(k+1)]$, being the composition of a convex function (the infinity norm) with linear functions (the state trajectory of the THPN and $\widetilde{m}$ between two consecutive switches). Thus, it attains its maximum either at $\tau(k)$ or at $\tau(k+1)$. Hence,

$$\begin{aligned}
&\max_{\tau\in[\tau(0),\tau(N)]} \|m(\tau)-\widetilde{m}(\tau)\|_\infty \\
&= \max_{0\le k\le N-1}\left\{\max_{\tau\in[\tau(k),\tau(k+1)]}\|m(t)-\widetilde{m}(t)\|_\infty\right\} \\
&= \max_{0\le k\le N-1}\{\max\{\|m(\tau(k))-\widetilde{m}(\tau(k))\|_\infty, \|m(\tau(k+1))-\widetilde{m}(\tau(k+1))\|_\infty\}\} \\
&= \max_{0\le k\le N}\{\|m(\tau(k))-\widetilde{m}(\tau(k))\|_\infty\} = \max_{k=0,\dots,\tau_N}\{\|m(k|\tau)-\widetilde{m}(k|\tau)\|_\infty\}
\end{aligned}$$

□

Note that cost function (27) still leads to a mixed-integer linear formulation of problem (19).

Remark: The cost function (24) searches for the minimum time trajectory from the initial marking to the desired marking using *at most* N events. Note in fact that the time to reach the desired marking is not a multiple of a sampling period (as it is for discrete time control) and that "fake" events, i.e., events with 0 time separation under which the state does not change, can be generated is less than N events are needed. The search for the optimal N makes the problem nonlinear, but it can be dealt with by iterative schemes similarly as in the discrete time case [17]. However, the possibility of "fake" events simplifies the search, as for large N the optimal solution is readily found.

4.2 Event-driven model predictive control

Problem (19) is a finite horizon open-loop optimal control problem, which computes the control profile $u(r)$, $r\in[\tau,\tau+\tau(N|\tau)]$, such that the constraints are satisfied and the cost is minimized. However, the control profile proceeds only for a finite number of events, where more events can be considered only at the price of an increased computational burden for solving (19). Furthermore, disturbances that occur during the execution of the control profile and possible modelling errors are not accounted for. Thus, a receding horizon feedback strategy is more advisable for cases where disturbances are possible. For this reason we incorporate the optimal control problem (19) in an event-driven closed-loop strategy based on Model Predictive Control (MPC) [8, 24].

The event-driven Model Predictive Control (eMPC) strategy operates as follows:

1. Let N be the event horizon; at a generic time τ set $\chi(\tau)=[m(\tau)'\ \psi(\tau)'\ \tau']'$.
2. Solve problem (19), to obtain the sequence of optimal controls $\mu^*(\chi(\tau)) = [\mu^*(0|\tau),\dots,\mu^*(N-1|\tau)]$.
3. Compute the input levels profile $\bar{u}_c^*(\chi(\tau)) = [u_c^*(0|\tau),\dots,u_c^*(N-1|\tau)] = \left[\frac{v^*(0|\tau)}{q^*(0|\tau)},\dots,\frac{v^*(N-1|\tau)}{q^*(N-1|\tau)}\right]$,
4. Apply $u(\xi)=[u_c^*(0|\tau)'\ u_d^*(0|\tau)']'$ for $\xi\in[\tau,\tau+q^*(0|\tau)]$.
5. Set $\tau=\tau+q^*(0|\tau)$, measure the new value of $\chi(\tau)$ and go to Step 2.

The actual state $m(\tau + q(0|\tau))$ at the end of each control action may be different from the predicted one $m(1|\tau)$ because of external disturbances and modelling errors. In fact, also the time instant at which the optimization problem is repeated may be different from the scheduled instant $\tau + q(0|\tau)$. By the closed-loop nature of the eMPC approach, the current state (and time) are measured or estimated again and a new updated optimal input sequence is computed.

For the nominal case, i.e., the trajectory is not perturbed by external disturbances, the reachability of a desired target marking can be proven.

Proposition 2 *Consider the event-driven MPC scheme applied to a THPN where the cost function is the minimum-time criterion* (24)*, and where the terminal constraint* (20) *is applied on the desired target marking $\widetilde{m}$. If the problem is feasible at time τ_0 with finite cost, then it is recursively feasible and the desired marking is reached in finite time, i.e., $m(\tau) = \widetilde{m}$ for $\tau < \infty$.*

Proof: The result follows from the convergence of the eMLD scheme proved in [13]. Since at time τ_0 problem (19) is feasible with finite cost, due to terminal constraint and minimum-time criterion, the command sequence $\mu^*(\tau_0)$ brings the marking to the target in finite time $J^*(\tau_0)$. A time $\tau_1 = \tau_0 + q^*(0|\tau_0)$, a new optimization problem is solved, where the sequence $[\mu^*(1|\tau_0), \ldots, \mu(N|\tau_0), \mu(N|\tau_1)]$ where $\mu(N|\tau_1) = [0\ 0\ 0]'$ is feasible, and brings the marking to the target in time $J(\tau_1) = J^*(\tau_0) - q^*(0|\tau_0)$. Hence, $J^*(\tau_1) \leq J^*(\tau_0) - q^*(0|\tau_0)$, and by recursive application, $J^*(\tau_k) + \sum_{i=0}^{k-1} q^*(0|\tau_0) \leq J^*(\tau_0)$, which means that the time computed at the first step it is always a lower bound for the time to reach the marking. Since $J^*(\tau_0) < \infty$ the time to reach the marking is finite. $\square$

Similar reachability results can be proved for the other criteria, where however the target may be reached only asymptotically in time, because convergence time is not explicitly accounted for in the cost function.

5 Case studies

This section presents two manufacturing systems modeled by hybrid Petri nets to which an event-driven MPC approach has been applied. The first system is a multiclass machine, the second one is a production network.

5.1 Multiclass machine

The hybrid Petri net in Figure 3 models a production system consisting of two lines and a single machine that processes the items in both lines. The first (second) line is modeled by transitions t_1, t_2 (t_3, t_4), and places p_1, p_3 (p_2, p_4), and has a capacity of c_1 (c_2) items. The input flows of the first and second lines are given by the flows of t_1 and t_3, respectively. Places p_1 and p_2 are the buffers to store the incoming parts from t_1 and t_3 before being processed. The output flow of the first (second) class is represented by t_2 (t_4). The processing machine is modeled by transitions t_5, t_6, t_7, t_8 and places p_5, p_6, p_7, p_8. Since the number of items in the lines is expected to be high, the places and transitions for the lines are continuous. On the other hand, since only one machine is available, the subnet that models the machine is discrete.

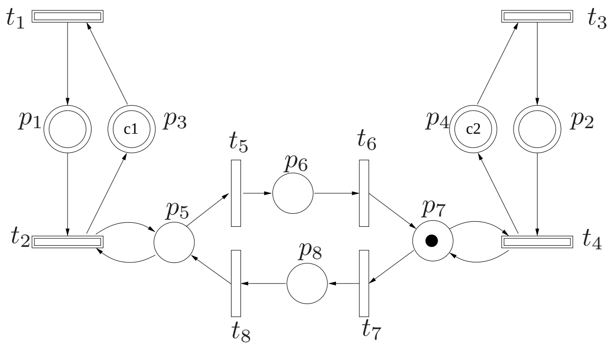


Figure 3: Two production lines and a multiclass machine.

Let the capacity of the buffers be $c_1 = 30$, $c_2 = 25$, and $\lambda[t_1] = 1.5$, $\lambda[t_2] = 2$, $\lambda[t_3] = 1$, $\lambda[t_4] = 3$. It is assumed that the processing machine needs 2 time units to change from one line to the other. During such 2 time units, none of the lines is processed. This is modeled by a deterministic delay of 2 units in transitions t_6 and t_8, i.e., $\vartheta[t_6] = \vartheta[t_8] = 2$, and by immediate transitions t_5 and t_7, i.e., $\vartheta[t_5] = \vartheta[t_7] = 0$. Let the initial marking of the system be $m_0[p_1] = 5$, $m_0[p_2] = 15$ and $m_0[p_7] = 1$. The marking of the remaining places is uniquely defined by these, since the invariants $m[p_1] + m[p_3] = c_1$, $m[p_2] + m[p_4] = c_2$ and $m[p_5] + m[p_6] + m[p_7] + m[p_8] = 1$ must hold.

It is required to compute a control law that maximizes the number of items produced over a given time interval. This is equivalent to maximizing the sum of the integral flows, v, of transitions t_2 and t_4. Hence, the cost function associated in (22) is defined by

$$F(m(N|\tau), \tau(N|\tau)) = 0, \quad L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = -(v[t_2](k|\tau) + v[t_4](k|\tau)). \tag{28}$$

Table 1 summarizes the obtained results for the described control problem for different prediction horizons N. Recall that the prediction horizon refers here to number of events. The control actions have been obtained by applying the event-driven MPC approach during a maximum time of 200 time units, i.e., the constraint $\tau(N|\tau_0) \leq 200$ has been added to (19d). In order to highlight the different performances of event-driven and standard MPC approaches, the same Table 1 also reports the control results obtained by standard MPC. The sampling time for standard MPC must guarantee that the time duration of the deterministic transitions t_6 and t_8 is 2, i.e., 2 must be a multiple of the sampling period. Notice that, the shorter the time period, the higher the performance that the standard MPC can achieve. This is because during the sampling period discrete transitions cannot be fired. For the sake of efficiency, the sampling period has been set to 2.

In Table 1, column N is the prediction horizon, column $\overline{v[t_2] + v[t_4]}$ is the average sum of flows, i.e., the sum of integral flows $v[t_2]$ and $v[t_4]$ per time unit, and *CPU Time* is the computational cost in seconds. All the experiments in this paper have been performed in Matlab 7.6.0.324(R2008a) environment running on a MacOS with 2.4 GHz Intel Core Duo and 4GB of RAM. It can be seen that for short prediction horizons, the event-driven performs better and its computational cost is lower. This is due to the fact that the time elapsed between events is variable and events happen only when necessary. On the other hand, although for very long prediction horizons, the event-driven approach still performs better, but its computational cost is higher than the one of the standard approach. The reason for this is that the resulting MILP for the event driven control contains more variables, e.g., the time elapsed between events, and therefore it scales worse than the MILP for standard MPC control. However, Table 1 shows that there is basically no need to go beyond $N = 6$, when eMPC still outperforms standard MPC also in terms of CPU time.

	Event-driven MPC		Standard MPC	
N	$\overline{v[t_2] + v[t_4]}$	CPU Time	$\overline{v[t_2] + v[t_4]}$	CPU Time
1	1.070	0.172	1.055	1.449
2	1.090	0.218	1.040	1.881
3	1.361	0.295	1.025	2.779
4	1.538	0.506	1.015	4.203
5	1.850	1.076	0.995	7.602
6	2.012	4.649	1.550	12.927
7	1.853	8.145	1.545	27.851
8	2.041	32.444	1.520	37.290
9	2.007	153.87	1.530	88.106
10	2.033	247.38	1.525	152.86

Table 1: Results of event-driven and standard MPC for different prediction horizons.

For the particular case of $N = 8$, the time evolution of the system under event-driven MPC control is shown in Figure 4. The state of the machine is shown by the line associated to *machine*: when the value is 2, the machine is processing line 1, i.e., $m[p_5] = 1$, when the value is 0, it is processing line 2, i.e., $m[p_7] = 1$, when the value is 1, it is swapping from one line to the other, i.e., either $m[p_6] = 1$ or $m[p_8] = 1$. It can be observed that the machine starts swapping as soon as one of the buffers becomes empty, i.e., for this particular set of parameters, the maximum production is obtained when the production of both lines is alternated. Note that, as expected for an event-driven formulation, a step only takes place when an event happens, and thus, in general, the duration of the steps is variable.

Let us now assume that input flows are subject to external uncontrollable and unknown disturbances that can modify the flow up to 10%. More precisely, this means that once the control action $u[t_1](u[t_3])$ is computed for a given interval, it is modified as $u[t_1] + \gamma \cdot u[t_1](u[t_3] + \gamma \cdot u[t_3])$, where γ is random variable with continuous uniform distribution in the interval $[-0.1, .1]$, before being applied to the system. Obviously, if the resulting value would produce negative markings it is truncated appropriately. Table 2 shows the performances and CPU time for both eMPC and standard MPC for several prediction horizons N. The same disturbances have been used in all cases. Similarly to the previous example without disturbances, it can be seen that, in general, eMPC performs better, but requires more CPU time as N gets higher.

Figure 5 shows the resulting trajectory of the system under event-driven MPC control with $N = 8$. The proposed eMPC strategy reacts to these perturbations by recomputing its control actions at each step.At the first step (around time instant 12), the marking of place p_1 is not as high as it would be if no perturbation existed. This is why in order to maximize the items produced over the

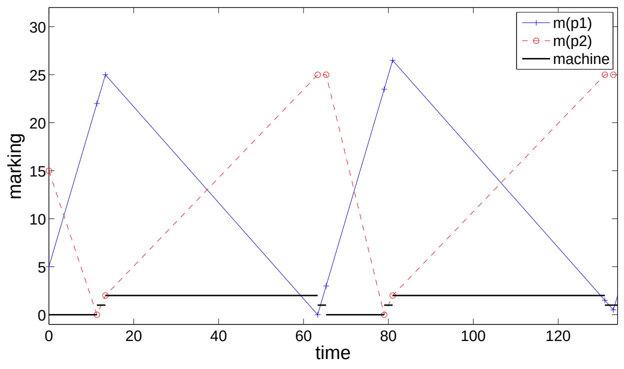


Figure 4: Time trajectory of the controlled system in Figure 3 without disturbances.

	Event-driven MPC		Standard MPC	
N	$v[t_2]+v[t_4]$	CPU Time	$v[t_2]+v[t_4]$	CPU Time
1	1.0333	0.158	1.0312	1.434
2	1.0588	0.205	1.0168	1.905
3	1.3563	0.293	1.0013	2.595
4	1.5145	0.563	0.9909	4.289
5	1.8075	1.379	1.4456	7.620
6	1.8483	3.857	1.4462	13.035
7	1.8972	10.577	1.5467	23.138
8	1.9962	32.679	1.5516	39.037
9	2.0515	179.00	1.5360	69.932
10	2.0189	323.35	1.5325	142.72

Table 2: Results of event-driven and standard MPC with disturbances for different prediction horizons.

specified period of time, the controller decides not to swap the machine to line 1 in order to let buffer of line 1 fill completely. It can be seen, that although the trajectory of the disturbed controlled system is slightly different to the nominal one (Figure 4), the controller manages to fill and empty buffers appropriately in order to maximize the performance.

5.2 Production network

In this section we consider the production network system described in [2]. The model of the system consists of continuous places and transitions representing buffers and flows of items, and two discrete places and transitions modeling a single machine, see Figure 6. In contrast to the model in [2] and in order to model the system more realistically, the net in Figure 6 includes complementary places for every buffer, so that the system is structurally bounded, and models the existing machine with a discrete subnet. In this model we assume that the time spent by the machine to swap from one line to the other is negligible, i.e., $\vartheta[t_7] = \vartheta[t_8] = 0$.

Let the capacity of the buffers be $c_1 = 8$, $c_2 = 6$, $c_3 = 4$, $c_4 = 8$, $c_5 = 6$, the upper bound of the transitions flows be $\lambda[t_1] = 0.5$, $\lambda[t_2] = 1.5$, $\lambda_[t_3] = 0.6$,

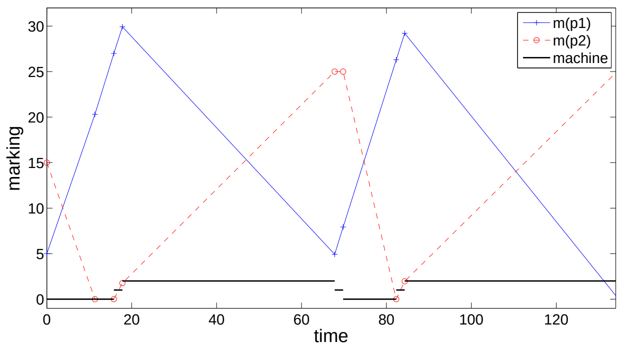


Figure 5: Time trajectory of the controlled system in Figure 3 with disturbances in flows of t_1 and t_3 (b).

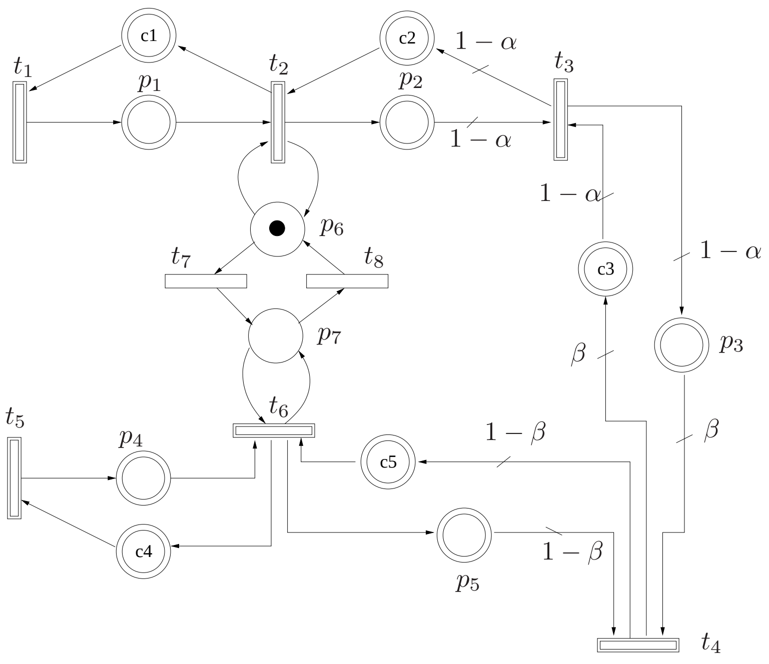


Figure 6: A production network modeled as a THPN.

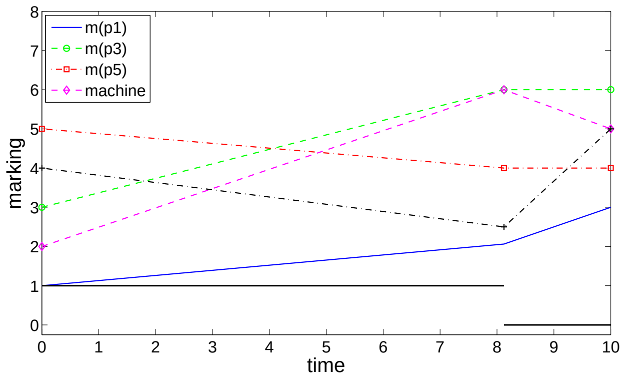
$\lambda[t_4] = 1$, $\lambda[t_5] = 0.8$, $\lambda[t_6] = 1.5$, and $\alpha = 0.1$, $\beta = 0.4$. Assume that the initial marking of the system is $m_0[p_1] = 1$, $m_0[p_2] = 3$, $m_0[p_3] = 5$, $m_0[p_4] = 2$, $m_0[p_5] = 4$ and $m_0[p_6] = 1$.

We first search for the minimum time control sequence to reach the target marking $m[p_1] = 3$, $m[p_2] = 6$, $m[p_3] = 4$, $m[p_4] = 5$, $m[p_5] = 5$, while no target marking is specified for the machine. After adding the constraint for the desired target marking (20), the objective function of the control problem is set to minimum time criterion

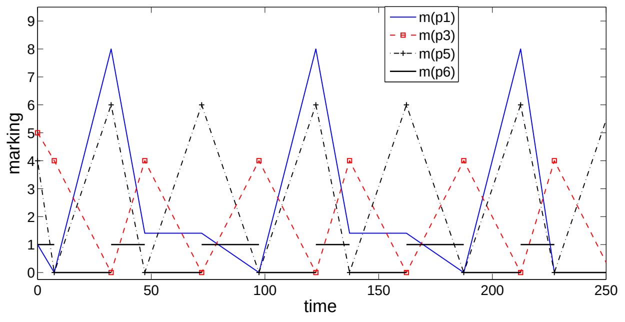
$$F(m(N|\tau), \tau(N|\tau)) = 0, \quad L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = q(k|\tau), \tag{29}$$

where we have set $N = 3$.

Figure 7(a) shows the time evolution of the system under the obtained control actions. The value associated to the label 'machine' indicates in which place the machine is located: if the value is 1 then the token is in p_6, if the value is 0 then the token is p_7. The target marking is reached at the second step. During the first interval of time, which lasts 8.1 time units, the machine is processing the items in buffer p_1, and in the second time interval it is processing the items in buffer p_4. The CPU time to compute the control actions was 0.045 seconds.



(a)



(b)

Figure 7: Time trajectory of the controlled system in Figure 6 to reach a target marking (a); to maximize the number of items produced by t_4 (b).

We want to maximize the number of items produced during the first 250 time units. The objective function associated to such control problem is $L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = -v[t_4](k)$. The trajectory of the system under the computed control actions for a prediction horizon of 3 steps is shown in Figure 7(b). It can be observed that after the first step a repetitive pattern develops in order minimize the objective function. In this case, the computation time was 0.621 seconds.

6 Conclusions

In this paper we have introduced an event-driven scheme for controlling timed hybrid Petri nets with the aim of maintaining the performance of continuous-time approaches and the computability of discrete-time ones. While the control action is finitely parametrized as in discrete-time models, hence allowing the application of optimization-based control algorithms, by selecting the events to include mode switches and constraints activation, the event-driven strategy enforces constraints and mode switches continuously in time, hence avoiding intersampling constraint violation and mode-mismatch errors.

By representing the hybrid Petri net in the proposed event-driven formalism, we have proposed a finite horizon open-loop optimal control problem that, using different control objectives, optimizes the dynamic behavior of the net. The problem has been used as the base of an event-driven model predictive control strategy that is a closed-loop control strategy and hence able to counteract the effect of external disturbances.

We have evaluated the behavior of the proposed algorithms on two examples obtained from the literature. One of the main issues to be investigated in the future is the consideration of other firing semantics, e.g, infinite server, both in discrete and continuous transitions.

References

[1] H. Alla and R. David. A modeling and analysis tool for discrete event systems: Continuous Petri net. *Performance Evaluation*, 33:175–199, 1998.

[2] F. Balduzzi, A. Giua, and G. Menga. First-Order Hybrid Petri Nets: a Model for Optimization and Control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.

[3] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming- the explicit solution. *IEEE Trans. Automatic Control*, 47(12):1974–1985, 2002.

[4] A. Bemporad and S. Di Cairano. Model predictive control of discrete hybrid stochastic automata. *IEEE Trans. Automatic Control*, 56(6):1307 –1321, 2011.

[5] A. Bemporad, S. Di Cairano, and J. Júlvez. Event-driven optimal control of integral continuous-time hybrid automata. In *Proc. 44th IEEE Conf. on Decision and Control*, pages 1409–1414, Seville, Spain, 2005.

[6] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[7] M.W. Braun, D.E. Rivera, M.E. Flores, W.M. Carlyle, and K.G. Kempf. A model predictive control framework for robust management of multi-product, multi-echelon demand networks. *Annual Reviews in Control*, 27(2):229–245, 2003.

[8] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 2004.

[9] Dash Associates. *XPRESS-MP User Guide*, 2003. `http://www.dashoptimization.com`.

[10] R. David and H. Alla. *Discrete, Continuous and Hybrid Petri Nets*. Springer-Verlag, 2010. 2^{nd} edition.

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

[12] S. Di Cairano. *Model Predictive Control of Hybrid Dynamical Systems: Stabilization, Event-driven, and Stochastic Control*. PhD thesis, Dip. Ing. Informazione, Universitá di Siena, Itali, 2008. `http://phd.dii.unisi.it/people/tesi/172_stefano_dicairano.pdf`.

[13] S. Di Cairano, A. Bemporad, and J. Júlvez. Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics. *Automatica*, 45(5):1243–1251, 2009.

[14] S. Di Cairano, D. Yanakiev, A. Bemporad, I.V. Kolmanovsky, and D. Hrovat. An MPC design flow for automotive control and applications to idle speed regulation. In *Proc. 47th IEEE Conf. on Decision and Control*, pages 5686–5691, Cancun, Mexico, 2008.

[15] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Trans. Contr. Systems Technology*, 15(3):566–580, 2007.

[16] A. Giua, C. Mahulea, L. Recalde, C. Seatzu, and M. Silva. Optimal control of timed continuous Petri nets via explicit MPC. *Lecture notes in control and information sciences*, 341:383–390, 2006.

[17] P. Grieder, M. Kvasnica, M. Baotic, and M. Morari. Low complexity control of piecewise affine systems with stability guarantee. In *American Control Conference, 2004. Proceedings of the 2004*, volume 2, pages 1196 – 1201 vol.2, June 2004.

[18] O. Gusikhin and E. Klampfl. Integrated process planning and supply chain configuration for commodity assemblies using petri nets. *Applications and Theory of Petri Nets*, pages 125–144, 2010.

[19] Ø. Hegrenæs, J.T. Gravdahl, and P. Tøndel. Spacecraft attitude control using explicit model predictive control. *Automatica*, 41(12):2107–2114, 2005.

[20] ILOG, Inc. *CPLEX 9.0 User Manual*. Gentilly Cedex, France, 2004.

[21] M. Johansson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.

[22] J. Júlvez, A. Bemporad, L. Recalde, and M. Silva. Event-Driven Optimal Control of Continuous Petri Nets. In *43rd IEEE Conference on Decision and Control (CDC)*, pages 69–74, Paradise Island. Bahamas, 2004.

[23] J.T. Linderoth and A. Lodi. Milp software. *Wiley Encyclopedia of Operations Research and Management Science*, 5:3239–3248, 2010.

[24] J.M. Maciejowski. *Predictive control with constraints.* Englewood Cliffs, NJ: Prentice Hall., 2002.

[25] C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. Optimal model predictive control of Timed Continuous Petri nets. *IEEE Trans. Automatic Control*, 53(7):1731–1735, 2008.

[26] A. Makhorin. *GLPK (GNU Linear Programming Kit) User's Guide*, 2003.

[27] D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: An LMI approach. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 504–509, December 2000.

[28] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[29] S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 93, no. 316:733–764, 2003.

[30] L. Recalde and M. Silva. Petri Nets Fluidification revisited: Semantics and Steady state. *APII-JESA*, 35(4):435–449, 2001.

[31] M. Silva. Introducing Petri Nets. *Practice of Petri Nets in Manufacturing, Chapman & Hall*, pages 1–62, 1993.

[32] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.

[33] J. Xu, L. Recalde, and M. Silva. Tracking Control of Join-Free Timed Continuous Petri Net Systems under Infinite Servers Semantics. *Discrete Event Dynamic Systems*, 18(2):263–283, 2008.