Approaching minimum time control of timed continuous Petri nets
-draft-

Hanife Apaydin-Özkan, Jorge Júlvez, C. Mahulea and Manuel Silva*

December 23, 2014

Abstract

This paper considers the problem of controlling timed continuous Petri nets under infinite server semantics. The proposed control strategy assigns piecewise constant flows to transitions in order to reach the target state. First, by using linear programming, a method driving the system from the initial to the target state through a linear trajectory is developed. Then, in order to improve the time of the trajectory, intermediate states are added by means of bilinear programming. Closed loop control is considered at the end. The algorithms developed here are then applied to a manufacturing system.

Published as:
H. Apaydin-Ozkan, J. Julvez, C. Mahulea, and M. Silva, "Approaching Minimum Time Control of Timed Continuous Petri nets," Nonlinear Analysis: Hybrid Systems, vol. 5, no. 2, pp. 136 - 148, May 2011. DOI: `http://doi.org/10.1016/j.nahs.2010.04.002`

*The authors are with Instituto de Investigación en Ingeniería de Aragon (I3A), Universidad de Zaragoza, Spain. E-mail: {*hapaydin, julvez, cmahulea, silva*}*@unizar.es.*
This work was partially supported by CICYT - FEDER projects DPI2006-15390, TIN2007-66523, and by the European Communitys Seventh Framework Programme under project DISC (Grant Agreement n. INFSO-ICT-224498). The work of H.Apaydin-Özkan was supported in part by the Diputacin General de Aragón for one year stay in the Group of Zaragoza; on leave from Faculty of Electrical and Electronics Engineering, Anadolu University, Eskisehir, Turkey.
This paper has been written in honor to Prof. L. Recalde, who passed away last December.

1 Introduction

Petri nets (PNs) are frequently used for modeling, analysis and synthesis of discrete event systems. The distributed state or marking of a PN is given by a vector of natural numbers which represent the number of tokens in each place. Similarly to most formalisms for discrete event systems, PNs suffer from the *state explosion problem.* One possible way to overcome this problem is *fluidification*, a classical relaxation technique.

Continuous Petri nets are a fluid approximation of classical discrete Petri nets ([1, 9]). Unlike conventional PNs, a transition in a continuous Petri net can be fired in a "real" non integer quantity. This leads to continuous markings instead of discrete ones. Similarly to discrete PNs, a time delay can be associated to the firing of transitions, the resulting net is called *timed continuous Petri net* (contPN).

As in discrete PNs, different server semantics can be adopted for the firing of transitions. Among them the most widely used semantics are *finite server* and *infinite server.* In [7], it is shown that infinite server semantics provides a better approximation of the steady state throughput than finite server semantics for a wide class of Petri nets under some general conditions.

A ContPN system with infinite server semantics is a subclass of *Piecewise Linear Systems* (PWL), with input constraints. A PWL system is composed of several linear subsystems together with switching rules. Many works deal with stability analysis or control law design of such systems ([8, 11, 2]). However, most of these methods cannot be directly applied to contPNs, because of their particular dynamic input constraints.

Control of contPNs with infinite server semantics has already been considered in the literature. In [6] it is shown that, the optimal steady state control problem of timed contPN system can be solved by means of *Linear Programming Problem* (LPP) when all transitions are assumed to be controllable. The transitory control problem is also solved by means of implicit and explicit MPC control strategy ([5]). A Lyapunov-function-based dynamic control algorithm was proposed in ([10]), which can ensure global convergence of both system states and input signals. In contrast to the method in this last reference, the technique proposed here for the computation of control law for direct linear trajectory is based on linear programming instead of nonlinear programming. Although the complexity of this new approach is significantly lower, i.e., it can be solved in polynomial time, the time of the obtained trajectory is very similar in general. Moreover, the method in ([10]) requires solving a *BiLinear Programming Problem* (BLP) for the computation of intermediate states as our method requires. Refinement of the trajectory by means of intermediate states is carried out by a recursive algorithm which computes new intermediate states until the time can not be improved significantly.

In this work, we design a control strategy which aims at driving the system from an initial state to a target one by minizing the time. The proposed strategy computes first the control actions to drive the system to the target state through a straight line. Such control actions are the result of solving a LPP. Then, as in [10], in order to obtain faster trajectories, intermediate states, not necessarily on the line connecting the initial and the target marking, are computed by means of a BLP. Moreover, an algorithm that recursively refines the initially obtained trajectory is given. These computations are then used to develop a closed loop scheme. Finally, a manufacturing system is considered and studied trying to obtain a number of products in minimum time.

This paper is structured as follows. Section 2 briefly introduces the required concepts of contPN systems and introduces the formulation of applied control. A LPP based method to obtain linear trajectories in order to reach target marking, $\boldsymbol{m}_f$, from initial marking $\boldsymbol{m}_0$ is given in Section 3. In Section 4, the heuristics for minimizing time by means of intermediate states is considered while Section 5 deals with closed loop approach. A manufacturing system is taken as a case study in Section 6. Finally, some conclusions and future directions are drawn in Section 7.

2 Continuous Petri nets

We assume that the reader is familiar with discrete PNs. In contPN systems, the firing is not restricted to the natural numbers but to the nonnegative real numbers.

2.1 Timed Continuous Petri nets

Definition 1 *A continuous PN system is a pair* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *where* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *is the net structure with the set of places* P*, the set of transitions* T*, pre and post matrices* $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{R}_{\geq 0}^{|P|\times|T|}$ *and* $\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$ *is the initial state.*

Let p_i, $i = 1, \ldots, |P|$ and $t_j, j = 1, \ldots, |T|$ denote the places and transitions. For a place $p_i \in P$ and a transition $t_j \in T$, Pre_{ij} and $Post_{ij}$ represent the weights of the arcs from p_i to t_j and from t_j to p_i, respectively. Each place p_i has a marking denoted by $m_i \in \mathbb{R}_{\geq 0}$. The vector of all token loads is called *state* or *marking*, and is denoted by $\boldsymbol{m} \in \mathbb{R}_{\geq 0}^{|P|}$. For every node $v \in P \cup T$, the sets of its input and output nodes are denoted as ${}^\bullet v$ and $v^\bullet$, respectively.

A transition $t_j \in T$ is enabled at $\boldsymbol{m}$ iff $\forall p_i \in^\bullet t_j$, $m_i > 0$ and its enabling degree is given by

$$enab(t_j, \boldsymbol{m}) = \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m_i}{Pre_{ij}} \right\}$$

which represents the maximum amount in which t_j can fire. An enabled transition t_j can fire in any real amount α, with $0 < \alpha \leq enab(t_j, \boldsymbol{m})$ leading to a new state $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}_{\cdot j}$ where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token flow matrix and $\boldsymbol{C}_{\cdot j}$ is its j^{th} column. If $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$ through a finite sequence σ, the state (or fundamental) equation is satisfied: $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$, where $\boldsymbol{\sigma} \in \mathbb{R}_{\geq 0}^{|T|}$ is the firing count vector, i.e., σ_j is the cumulative amount of firings of t_j in the sequence σ.

Definition 2 *A contPN system* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *is a continuous PN system* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *together with a vector* $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^{|T|}$ *where* λ_j *is the firing rate of transition* t_j*.*

The state equation has an explicit dependence on time, denoted by τ: $\boldsymbol{m}(\tau) = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$ which through time differentiation becomes $\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$. The derivative of the firing sequence $\boldsymbol{f}(\tau) = \dot{\boldsymbol{\sigma}}(\tau)$ is called the firing flow. Depending on how the flow is defined, many firing semantics appear, being the most used ones *infinite* and *finite* server semantics. This paper deals with infinite server semantics for which the flow of a transition t_j is defined as:

$$f_j(\tau) = \lambda_j \cdot enab(t_j, \boldsymbol{m}(\tau)) = \lambda_j \cdot \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m_i(\tau)}{Pre_{ij}} \right\} \tag{1}$$

Since the flow of a transition depends on its enabling degree which is based on the minimum function, a timed contPN with infinite server semantics is a PWL system with polyhedral regions and everywhere continuous vector field. The state space (or reachability set) $\mathcal{R}$ of a contPN system can be divided into regions[1] as follows: $\mathcal{R} = \mathcal{R}^1 \cup ... \cup \mathcal{R}^\gamma$ where $\gamma \leq \prod_{j=1}^{|T|} |{}^\bullet t_j|$. Intuitively, each $\mathcal{R}^z$ denotes a region where the flow is limited by the same subset of places (one for each transition). More formally, two markings m_a and m_b belong to the same region $\mathcal{R}^z$ iff $[\forall\, t \in T$ and $\forall\, p_u, p_v \in {}^\bullet t$ it holds that $m_a(p_u) \leq m_a(p_v) \leftrightarrow m_b(p_u) \leq m_b(p_v)]$.

For a given $\mathcal{R}^z$, we can define the constraint matrix $\boldsymbol{\Pi}^z \in \mathbb{R}^{|P|}_{\geq 0}$ as follows. Let $\boldsymbol{m}$ be an interior point of $\mathcal{R}^z$, then $\boldsymbol{\Pi}^z$ is defined as:

$$\boldsymbol{\Pi}^z_{ji} = \begin{cases} \frac{1}{Pre_{ij}}, & if \;\; \frac{m_i}{Pre_{ij}} = \min_{p_h \in {}^\bullet t_j} \left\{ \frac{m_h}{Pre_{hj}} \right\} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

If marking $\boldsymbol{m}$ belongs to $\mathcal{R}^z$, we denote $\boldsymbol{\Pi}(\boldsymbol{m}) = \boldsymbol{\Pi}^z$ the corresponding constraint matrix (if $\boldsymbol{m}$ is a border point and belongs to several regions, $\boldsymbol{\Pi}(\boldsymbol{m})$ is the constraint matrix of any of these regions). In the constraint matrix, each row $j = 1, 2..., |T|$ has only one non-null element in the position i that corresponds to the place p_i that restricts the flow of transition t_j. Furthermore, the firing rate of transitions can also be represented by a diagonal matrix $\boldsymbol{\Lambda} = \text{diag}\{\lambda_1, ...\lambda_{|T|}\}$. Using this notation, the nonlinear flow of the transitions at a given state $\boldsymbol{m}$ can be written as:

$$\boldsymbol{f} = \boldsymbol{\Lambda} \cdot \boldsymbol{\Pi}(\boldsymbol{m}) \cdot \boldsymbol{m} \tag{3}$$

Example 3 *Let us consider the PN in Fig. 1. Assume* $\boldsymbol{\lambda} = [4\ 1\ 3\ 1]^T$, $\boldsymbol{m}_0 = [7.5\ 4.5\ 4\ 2\ 1.5\ 5\ 4\ 2.5]^T$ *and* $\boldsymbol{m}_f = [7\ 5\ 5\ 1\ 1\ 4\ 5\ 3]^T$.

Independently of $\boldsymbol{m}_0$*, the system dynamics is described as follows:*

$$\begin{array}{rcl} \dot{m}_1 &=& f_4 - f_1 = \lambda_4 \cdot m_4 - \lambda_1 \cdot \min\{m_5, m_1, m_8\} \\ \dot{m}_2 &=& f_1 - f_2 = \lambda_1 \cdot \min\{m_5, m_1, m_8\} - \lambda_2 \cdot \min\{m_2, m_6\} \\ \dot{m}_3 &=& f_2 - f_3 = \lambda_2 \cdot \min\{m_2, m_6\} - \lambda_3 \cdot \min\{m_3, m_7, m_8\} \\ \dot{m}_4 &=& f_3 - f_4 = \lambda_3 \cdot \min\{m_3, m_7, m_8\} - \lambda_4 \cdot m_4 \\ \dot{m}_5 &=& f_2 - f_1 = \lambda_2 \cdot \min\{m_2, m_6\} - \lambda_1 \cdot \min\{m_5, m_1, m_8\} \\ \dot{m}_6 &=& f_3 - f_2 = \lambda_3 \cdot \min\{m_3, m_7, m_8\} - \lambda_2 \cdot \min\{m_2, m_6\} \\ \dot{m}_7 &=& f_4 - f_3 = \lambda_4 \cdot m_4 - \lambda_3 \cdot \min\{m_3, m_7, m_8\} \\ \dot{m}_8 &=& f_2 + f_4 - f_1 - f_3 = \lambda_2 \cdot \min\{m_2, m_6\} + \lambda_4 \cdot m_4 \\ && -\lambda_1 \cdot \min\{m_5, m_1, m_8\} - \lambda_3 \cdot \min\{m_3, m_7, m_8\} \end{array} \tag{4}$$

[1]These regions are disjoint except possibly on the borders.

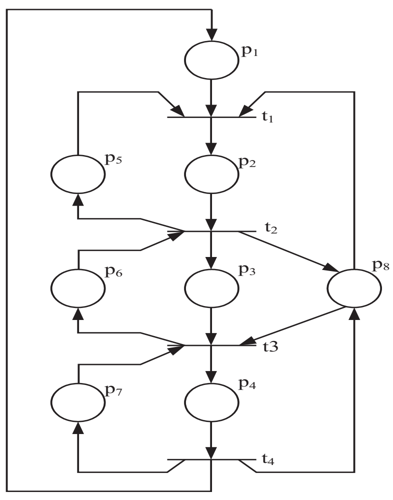


Figure 1: A PN taken from ([12])

Note that $\boldsymbol{m}_0$ *and* $\boldsymbol{m}_f$ *are in different regions:* $\boldsymbol{m}_0 \in \mathcal{R}^1 : \{\boldsymbol{m} \mid m_5 \leq m_1,\ m_5 \leq m_8,\ m_8 \leq m_7,\ m_8 \leq m_3,\ m_2 \leq m_6\}$ *and* $\boldsymbol{m}_f \in \mathcal{R}^2 : \{\boldsymbol{m} \mid m_5 \leq m_1,\ m_5 \leq m_8,\ m_8 \leq m_7,\ m_8 \leq m_3,\ m_6 \leq m_2\}$*.*

The system dynamics for $\mathcal{R}^1$ *are described as:*

$$\left\{\begin{array}{lcl} \dot{m_1} & = & m_4 - 4 \cdot m_5 \\ \dot{m_2} & = & 4 \cdot m_5 - m_2 \\ \dot{m_3} & = & m_2 - 3 \cdot m_8 \\ \dot{m_4} & = & 3 \cdot m_8 - m_4 \\ \dot{m_5} & = & m_2 - 4 \cdot m_5 \\ \dot{m_6} & = & 3 \cdot m_8 - m_2 \\ \dot{m_7} & = & m_4 - m_8 \\ \dot{m_8} & = & m_2 + m_4 - 4 \cdot m_5 + 3 \cdot m_8 \end{array}\right. \tag{5}$$

While the system dynamics for $\mathcal{R}^2$ *are:*

$$\left\{\begin{array}{lcl} \dot{m_1} & = & m_4 - 4 \cdot m_5 \\ \dot{m_2} & = & 4 \cdot m_5 - m_6 \\ \dot{m_3} & = & m_6 - 3 \cdot m_8 \\ \dot{m_4} & = & 3 \cdot m_8 - m_4 \\ \dot{m_5} & = & m_6 - 4 \cdot m_5 \\ \dot{m_6} & = & 3 \cdot m_8 - m_6 \\ \dot{m_7} & = & m_4 - m_8 \\ \dot{m_8} & = & m_6 + m_4 - 4 \cdot m_5 + 3 \cdot m_8 \end{array}\right. \tag{6}$$

2.2 Controlled Timed Continuous Petri Nets

In this section we consider contPN systems subject to external control actions, and assume that the only admissible control law consists in *slowing-down* the firing speed of transitions ([9]). If a transition can be controlled (its flow can be reduced or even stopped), we will say that it is a controllable transition ([6]).

Definition 4 *The controlled flow, $\boldsymbol{w}$, of a timed contPN is defined as $\boldsymbol{w}(\tau) = \boldsymbol{f}(\tau) - \boldsymbol{u}(\tau)$, with $0 \leq \boldsymbol{u}(\tau) \leq \boldsymbol{f}(\tau)$, where $\boldsymbol{f}$ is the flow of the uncontrolled system, i.e., defined as in (1), and $\boldsymbol{u}$ is the control action.*

Therefore, the control input $\boldsymbol{u}$ is dynamically upper bounded by the flow $\boldsymbol{f}$ of the corresponding unforced system. Under these conditions, the overall behaviour of the system in which all transitions are controllable is ruled by the following system:

$$\begin{array}{l} \dot{\boldsymbol{m}} = \boldsymbol{C} \cdot [\boldsymbol{f} \; - \; \boldsymbol{u}] = \boldsymbol{C} \cdot \boldsymbol{w} \\ 0 \leq \boldsymbol{u} \leq \boldsymbol{f} \end{array} \tag{7}$$

The constraint $0 \leq \boldsymbol{u} \leq \boldsymbol{f}$ can be rewritten as $0 \leq \boldsymbol{f} - \boldsymbol{u} \leq \boldsymbol{f}$. Defining $\boldsymbol{w} = \boldsymbol{f} - \boldsymbol{u}$ and using (3), the constraint can be expressed as:

$$0 \;\leq\; \boldsymbol{w} \;\leq\; \boldsymbol{\Lambda} \cdot \boldsymbol{\Pi}(\boldsymbol{m}) \cdot \boldsymbol{m} \tag{8}$$

The following sections show how to compute a control action $\boldsymbol{u}$ that drives the system from the initial marking $\boldsymbol{m}_0$ to a desired target marking $\boldsymbol{m}_f$. Section 3 describes a method to obtain a linear trajectory. Section 4 makes use of such a method to compute piecewise linear trajectories in order to improve the time to reach $\boldsymbol{m}_f$. Notice that in order to be reachable, $\boldsymbol{m}_f$ necessarily satisfies the state equation, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. Our procedure consists of assigning constant controlled flow $\boldsymbol{w}$ that satisfies dynamic upper bounds. We assume that $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are interior points of the reachability space, i.e., the markings are strictly positive. The assumption that $\boldsymbol{m}_0$ is positive ensures that the system can move at $\tau = 0$ in the direction of $\boldsymbol{m}_f$; the assumption that $\boldsymbol{m}_f$ is positive ensures that $\boldsymbol{m}_f$ can be reached ([3]) in finite time ([6]).

3 Computation of Linear Trajectories

In this section, we will distinguish two cases: (A) $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are in the same region and (B) they are in different regions. In this last case the crossing points of the straight line and borders must be considered.

(A) $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are in the same region $\mathcal{R}^z$. Then all the states on the straight line connecting them are also interior points of $\mathcal{R}^z$ since all the regions are convex. The following LPP computes the correspondig control law, where $\boldsymbol{x} = \boldsymbol{w} \cdot \tau_f$,

$$\begin{array}{lll} \min & \tau_f & \\ & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{x} & (a) \\ & 0 \leq x_j \leq \lambda_j \cdot \Pi^z_{ji} \cdot \min\left\{m_{0i}, m_{f\,i}\right\} \cdot \tau_f, & \\ & \quad \forall\, j \in \{1, .., |T|\} \text{ where } i \text{ satisfies } \Pi^z_{ji} \neq 0 & (b) \end{array} \tag{9}$$

The equations correspond to: (a) the time dependent equation of the straight line connecting $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$, (b) flow constraints in (8). Notice that (9)(b) is a linear constraint because m_{0_i} and m_{f_i} are known.

(B) $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are in different regions. The line connecting $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ can be divided in several segments, each one starting in a border (or in $\boldsymbol{m}_0$ for the first segment) and ending in a border (or in $\boldsymbol{m}_f$ for the last one). Then LPP (9) is applied on each of these segments.

Algorithm 1 computes the total time τ_f by using LPP in (9) for the linear trajectory ℓ from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ which crosses $n \in \{1, 2, .., \gamma\}$ borders. In this algorithm, $\boldsymbol{m}_c^i$ stands for state at the intersection of ℓ and i^{th} crossed border (starting from $\boldsymbol{m}_0$) $i \in \{1, 2, .., n\}$. $\boldsymbol{m}_c^0$ and $\boldsymbol{m}_c^{n+1}$ denote $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$, respectively. $\boldsymbol{w}^i$ is the flow obtained from the state $\boldsymbol{m}_c^i$ to the state $\boldsymbol{m}_c^{i+1}$. Note that, if ℓ does not cross any border, then $n = 0$, that is $\boldsymbol{m}_c^0 = \boldsymbol{m}_0$ and $\boldsymbol{m}_c^1 = \boldsymbol{m}_f$.

Algorithm 1 Linear trajectory

```
Input: $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$,  $\boldsymbol{m}_f$
Compute the line $\ell$ connecting $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$
Compute the intersection of $\ell$ and the crossed borders: $\boldsymbol{m}_c^1$, $\boldsymbol{m}_c^2$, .....$\boldsymbol{m}_c^n$
$\boldsymbol{m}_c^0 = \boldsymbol{m}_0$,  $\boldsymbol{m}_c^{n+1} = \boldsymbol{m}_f$
for $i = 0$ to $n$ do
  Determine $\tau_i$ by solving LPP in (9) with $\boldsymbol{m}_0 = \boldsymbol{m}_c^i$ and $\boldsymbol{m}_f = \boldsymbol{m}_c^{i+1}$
end for
Output: $\tau_1, \boldsymbol{w}^1, \tau_2, \boldsymbol{w}^2, ..., \tau_{n+1}, \boldsymbol{w}^{n+1}$, $\tau_f = \sum_{i=1}^{n+1} \tau_i$
```

Proposition 5 *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \ \boldsymbol{m}_0 \rangle$ be a contPN system with $\boldsymbol{m}_0 > 0$. If $\boldsymbol{m}_f$ belongs to $\mathcal{R}$ and $\boldsymbol{m}_f > 0$:*

- *LPP(9) is feasible*
- *The control action given by Algorithm 1 ensures that $\boldsymbol{m}_f$ is reached in finite time.*

Proof: Since $\boldsymbol{m}_f$ is a reachable marking, then there exists $\boldsymbol{x}$ such that the state equation is satisfied 9(a). By taking τ_f sufficiently large 9(b) can be satisfied since $\lambda_j \cdot \Pi_{ji}^z \cdot \min\left\{m_{0i}, m_{f\,i}\right\} > 0$. Then, by ([3]) the linear trajectory from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ can be followed by the system since $\boldsymbol{m}_0 > 0$ and $\boldsymbol{m}_f > 0$, i.e. , all intermediate states are not spurious. Moreover, since $\boldsymbol{m}_f > 0$, by ([6]) $\boldsymbol{m}_f$ can be reached in finite time. □

Example 6 *Let us consider the control law design for the contPN in Ex. 3. Assume the same $\boldsymbol{\lambda}$, $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$. Note that the line ℓ connecting $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ crosses only one border: $m_2 = m_6$ and the crossing point is calculated as: $\boldsymbol{m}_c^1 = [7.33\ 4.67\ 4.33\ 1.67\ 1.33\ 4.67\ 4.33\ 2.67]^T$, $\boldsymbol{m}_c^0 = \boldsymbol{m}_0$, $\boldsymbol{m}_c^2 = \boldsymbol{m}_f$. The trajectory is illustrated in Fig.2. According to Algorithm 1, first LPP (9) with*

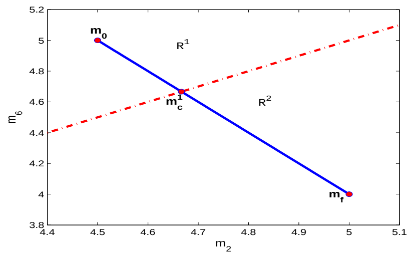


Figure 2: Trajectory for Ex. 6

$\boldsymbol{m}_0 = \boldsymbol{m}_c^0$ and $\boldsymbol{m}_f = \boldsymbol{m}_c^1$ is solved. Its constraints are:

$$\begin{array}{ll} 7.33 & = x_4 - x_1 + 7.5 \\ 4.67 & = x_1 - x_2 + 4.5 \\ 4.33 & = x_2 - x_3 + 4 \\ 1.67 & = x_3 - x_4 + 2 \\ 1.33 & = x_2 - x_1 + 1.5 \\ 4.67 & = x_3 - x_2 + 5 \\ 4.33 & = x_4 - x_3 + 4 \\ 2.67 & = x_2 + x_4 - x_1 - x_3 + 2.5 \end{array} \qquad (a)$$

$$\begin{array}{ll} 0 & \leq x_1 \leq 4 \cdot 1.33 \cdot \tau_f \\ 0 & \leq x_2 \leq 4.5 \cdot \tau_f \\ 0 & \leq x_3 \leq 3 \cdot 2.5 \tau_f \\ 0 & \leq x_4 \leq 1.66 \cdot \tau_f \end{array} \qquad (b)$$

The optimal solution is $\tau_f = 0.2$ t.u. and $w_1 = 2.49$, $w_2 = 1.67$, $w_3 = 0$, $w_4 = 1.67$. In the second case, i.e., $\boldsymbol{m}_0 = \boldsymbol{m}_c^1$ and $\boldsymbol{m}_f = \boldsymbol{m}_c^2$:

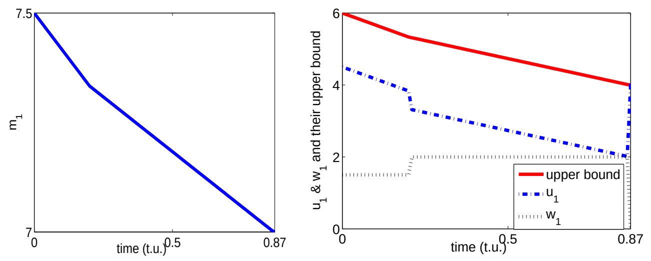
$$\begin{array}{ll} 7 & = x_4 - x_1 + 7.33 \\ 5 & = x_1 - x_2 + 4.67 \\ 5 & = x_2 - x_3 + 4.33 \\ 1 & = x_3 - x_4 + 1.67 \\ 1 & = x_2 - x_1 + 1.33 \\ 4 & = x_3 - x_2 + 4.67 \\ 5 & = x_4 - x_3 + 4.33 \\ 3 & = x_2 + x_4 - x_1 - x_3 + 2.67 \end{array} \qquad (a)$$

$$\begin{array}{ll} 0 & \leq x_1 \leq 4 \cdot \tau_f \\ 0 & \leq x_2 \leq 4 \cdot \tau_f \\ 0 & \leq x_3 \leq 3 \cdot 2.67 \cdot \tau_f \\ 0 & \leq x_4 \leq \tau_f \end{array} \qquad (b)$$

The optimal solution is $\tau_f = 0.67$ t.u. and $w_1 = 1.5$, $w_2 = 1$, $w_3 = 0$, $w_4 = 1$. The total time to go from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ through the line ℓ is $\tau_{total} = 0.2 + 0.67 = 0.87$ t.u. And the control is obtained as:

$$\boldsymbol{u}(\tau) = \begin{cases} \begin{bmatrix} 4 \cdot m_5(\tau) - 2.49 \\ m_2(\tau) - 1.67 \\ 3 \cdot m_8(\tau) \\ m_4(\tau) - 1.67 \end{bmatrix} & , \textit{if} \;\; 0 \leq \tau \leq 0.2 \\ \\ \begin{bmatrix} 4 \cdot m_5(\tau) - 1.5 \\ m_2(\tau) - 1 \\ 3 \cdot m_8(\tau) \\ m_4(\tau) - 1 \end{bmatrix} & , \textit{if} \;\; 0.2 < \tau \leq 0.87 \end{cases} \qquad (10)$$

Fig. 3(a) illustrates the convergence of the marking m_1 under the designed control law, while Fig. 3(b) shows the control signal u_1, w_1 and their upper bound.



(a) (b)

Figure 3: Evolution of (a) m_1 and (b) u_1, w_1 and and their upper bound for Ex. 6

4 A Heuristics for minimum time control

This section illustrates how to compose a piecewise linear trajectory by introducing intermediate states in order to improve the time duration to reach $\boldsymbol{m}_f$.

The designed control in the previous subsection takes the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ through a linear trajectory by minimizing the time duration. In some cases, i.e., when initial or final state have a small value, this may limit the speed of the response. In order to improve the time spent to move from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$, intermediate states denoted by $\boldsymbol{m}^k$ where $k \in \{1, 2...s\}$, that do not necessarily lie on the line from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$, are computed. The new trajectory is obtained as the union of $s+1$ segments: $\boldsymbol{m}_0 \to \boldsymbol{m}^1 \to \boldsymbol{m}^2 \to ... \to \boldsymbol{m}^s \to \boldsymbol{m}_f$. In this work we consider the intermediate states in the range:

$$\mathcal{R}^I = \Big\{ \boldsymbol{m} \mid m_i \leq \max\{m_{f_i}, m_{0_i}\}, \quad m_i \geq \min\{m_{f_i}, m_{0_i}\}, \\ i \in \{1, 2, .., |P|\} \Big\}, \tag{11}$$

Because $\boldsymbol{m}_0,\ \boldsymbol{m}_f > 0$, all $\boldsymbol{m} \in \mathcal{R}^I$ are positive and reachable ([3]). We will describe how to compute intermediate states that lie on the borders and the interior of regions, separately.

4.1 Intermediate states on the borders

Let the line from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ cross s borders, and pass through $s+1$ different regions: $\mathcal{R}^1$, $\mathcal{R}^2$, ..., $\mathcal{R}^{s+1}$ with corresponding constraint matrices $\boldsymbol{\Pi}^1$, $\boldsymbol{\Pi}^2$...$\boldsymbol{\Pi}^{s+1}$. We assign constant flows for each transitions during each line segments and the intermediate states on each border: $\boldsymbol{m}^1$, $\boldsymbol{m}^2$....$\boldsymbol{m}^s$. Under these assumptions, the following BLP with $\boldsymbol{m}^0 = \boldsymbol{m}_0$ and $\boldsymbol{m}^{s+1} = \boldsymbol{m}_f$ and with the variables $\boldsymbol{m}^1$, $\boldsymbol{m}^2$,....,$\boldsymbol{m}^{s-1}$, $\boldsymbol{m}^s$, τ_k , $\quad \boldsymbol{x}^k = \boldsymbol{w}^k \cdot \tau_k$, $k \in \{1, ...s\}$ obtains the intermediate states that yield a minimum time trajectory:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{s+1} \tau_k \\
& \boldsymbol{m}^{k+1} = \boldsymbol{m}^k + \boldsymbol{C} \cdot \boldsymbol{x}^{k+1}, \qquad k \in \{0,1,2,..,s\} && (a) \\
& \left(\boldsymbol{\Pi}^k - \boldsymbol{\Pi}^{k+1}\right) \cdot \boldsymbol{m}^k = 0, \qquad k \in \{1,2,..,s\} && (b) \\
& m_i^k \leq m_i^{k+1} \;\text{ if }\; m_{0_i} \leq m_{f_i} \\
& \qquad\qquad i \in \{\,1,2..|P|\}, \;\; k \in \{0,1,2,..,s\} && (c) \\
& m_i^k \geq m_i^{k+1} \;\text{ if }\; m_{0_i} \geq m_{f_i} \\
& \qquad\qquad i \in \{\,1,2..|P|\}, \;\; k \in \{0,1,2,..,s\} && (d) \\
& 0 \leq x_j^k \leq \lambda_j \cdot \Pi_{ji}^k \cdot \min\{m_i^{k-1},\; m_i^k\} \cdot \tau_k, \\
& \qquad \text{with } p_i \text{ st. } \Pi_{ji}^k \neq 0,\; j \in \{1,2...|T|\}, \;\; k \in \{1,2..s\} && (e)
\end{aligned}
\tag{12}
$$

The constraints correspond to (a) the time dependent equation of the connecting $\boldsymbol{m}^k$ to $\boldsymbol{m}^{k+1}$: $\boldsymbol{m}^{k+1} = \boldsymbol{m}^k + \boldsymbol{C} \cdot \boldsymbol{w}^{k+1} \cdot \tau_f$ and $\boldsymbol{m}^k$ is a reachable marking; (b) $\boldsymbol{m}^k$ is on the crossed border; (c & d) $\boldsymbol{m}^k$ is contained in the hypercube defined by the corners $\boldsymbol{m}^{k-1}$ and $\boldsymbol{m}^{k+1}$; (e) the constraints on the controlled flow.

4.2 Interior intermediate states

In the case that $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are in the same region $\mathcal{R}^z$ a recursive method can be used to determine intermediate states. The same recursive method can also be applied to find additional intermediate states between $\boldsymbol{m}^k$ and $\boldsymbol{m}^{k+1}$, since they are on the borders of the same region. In this method we keep computing intermediate states until the time cannot be improved by a considerable amount. In this case, the BLP that will be solved in each step can be simplified to the following problem where $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ are known; τ_1, τ_2, $\boldsymbol{m}^d$, $\boldsymbol{x}^1 = \boldsymbol{w}^1 \cdot \tau_1, \boldsymbol{x}^2 = \boldsymbol{w}^2 \cdot \tau_2$ are variables:

$$
\begin{aligned}
\min \quad & \tau_1 + \tau_2 \\
& \boldsymbol{m}^d = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{x}^1 \\
& \boldsymbol{m}_f = \boldsymbol{m}^d + \boldsymbol{C} \cdot \boldsymbol{x}^2, && (a) \\
& \boldsymbol{m}^d = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \;\; \boldsymbol{m}^d,\; \boldsymbol{\sigma} \geq 0 && (b) \\
& \min\{m_{f_i}, m_{0_i}\} \leq m_i^d \leq \max\{m_{f_i}, m_{0_i}\}, \\
& \qquad\qquad \forall i \in \{1,2...|P|\} && (c) \\
& 0 \leq x_j^1 \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min\{m_{0_i}, m_i^d\} \cdot \tau_1, \\
& \qquad \text{with } i \text{ st. } \Pi_{ji}^k \neq 0,\;\; j \in \{1,2...|T|\} \\
& 0 \leq x_j^2 \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min\{m_i^d, m_{f_i}\} \cdot \tau_2, \\
& \qquad \text{with } i \text{ st. } \Pi_{ji}^k \neq 0,\;\; j \in \{1,2...|T|\} && (d)
\end{aligned}
\tag{13}
$$

Algorithm 2 expresses the recursive method we proposed. It computes new intermediate states until the stopping criterion is satisfied. It is satisfied when the time of the trajectory cannot be decreased more than a given value ϵ. In the algorithm, *path* is a global variable storing the ordered set of couples of states in the trajectory and the relative times to reach them. The number of states in *path* is denoted by $size(path)$ and the i^{th} element of the set *path* is denoted by $path(i)$.

Algorithm 2 Piecewise Linear Trajectory

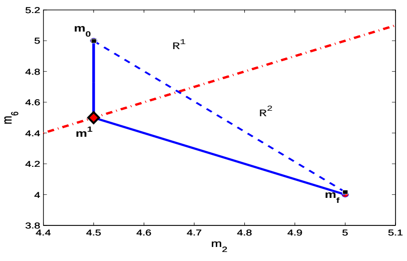
```
Input: $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$,  $\boldsymbol{m}_f$,  $\epsilon$
Global variable: $path$
Compute the line $\ell$ connecting $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$
Determine the number of crossed borders: $s$
if $s = 0$ then
   Solve (9) to obtain $\tau_f$
   $path_0 = \{(\boldsymbol{m}_0, 0),\ (\boldsymbol{m}_f, \tau_f)\}$
else
   Solve (12) to obtain $\tau_f$
   $path_0 = \{(\boldsymbol{m}_0, 0),\ (\boldsymbol{m}^1, \tau_1), ..., (\boldsymbol{m}_f, \tau_f)\}$
end if
$\tau_x = \tau_f$  $, path = path_0$
for $i = 1 : size(path_0) - 1$ do
   $(\boldsymbol{m}^x, \tau_x) = path_0(i)$  ,   $(\boldsymbol{m}^y, \tau_y) = path_0(i+1)$
   Call Split function with $((\boldsymbol{m}^x, \tau_x),\ (\boldsymbol{m}^y, \tau_y))$
end for
Calculate the total time for $path$, i.e., $\tau_y$
Output:$path$
```

Split function

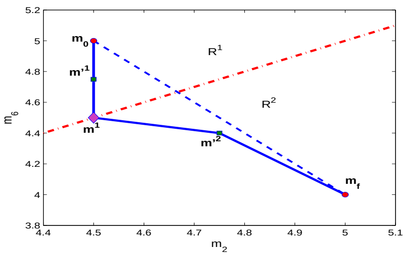
```
Input: $(\boldsymbol{m}^x, \tau_x),\ (\boldsymbol{m}^y, \tau_y)$
Solve (13) to obtain $\tau_1,\ \tau_2,\ \boldsymbol{m}^d$
if $\tau_y - (\tau_1 + \tau_2) > \epsilon$ then
   Insert $(\boldsymbol{m}^d, \tau_1)$ in $path$
   Change $(\boldsymbol{m}^y, \tau_y)$ in $path$ by $(\boldsymbol{m}^y, \tau_2)$
   Call Split function with $((\boldsymbol{m}^x, \tau_x),\ (\boldsymbol{m}^d, \tau_1))$
   Call Split function with $((\boldsymbol{m}^d, \tau_1),\ (\boldsymbol{m}^y, \tau_2))$
end if
```

Example 7 *Let us consider the contPN system considered in Ex. 6, again. Since the linear trajectory between $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ crosses only one border, $s = 1$ and solving (12) with $s = 1$ yields 0.83 t.u. as a total time. And the new path is $\boldsymbol{m}_0 \rightarrow \boldsymbol{m}^1 \rightarrow \boldsymbol{m}_f$, where $\boldsymbol{m}^1 = [7.5\ 4.5\ 4.5\ 1.5\ 1.5\ 4.5\ 4.5\ 3]^T$. For an additional intermediate state, say $\boldsymbol{m'}^1$, between $\boldsymbol{m}_0$ and $\boldsymbol{m}^1$, we solve (13) for $\boldsymbol{m}_0 = [7.5\ 4.5\ 4\ 2\ 1.5\ 5\ 4\ 2.5]^T$ and $\boldsymbol{m}_f = \boldsymbol{m}^1$. Similarly, for an additional intermediate state between $\boldsymbol{m}^1$ and $\boldsymbol{m}_f$, say $\boldsymbol{m'}^2$, we solve (13) for $\boldsymbol{m}_0 = \boldsymbol{m}^1$ and $\boldsymbol{m}_f = [7\ 5\ 5\ 1\ 1\ 4\ 5\ 3]^T$. And the new path $\boldsymbol{m}_0 \rightarrow \boldsymbol{m'}^1 \rightarrow \boldsymbol{m}^1 \rightarrow \boldsymbol{m'}^2 \rightarrow \boldsymbol{m}_f$ is obtained, where $\boldsymbol{m'}^1 = [7.5\ 4.5\ 4.26\ 1.74\ 1.5\ 4.74\ 4.26\ 2.76]^T$ and $\boldsymbol{m'}^2 = [7.45\ 4.78\ 4.55\ 1.22\ 1.22\ 4.45\ 4.78\ 3]^T$. The total time is reduced to 0.74 t.u. with the control action $u(\tau) = [4 \cdot m_5(\tau) - 1.74\ \ m_2(\tau) - 1.74\ \ 3 \cdot m_8(\tau)\ \ m_4(\tau) - 1.74]^T$*

for $0 \leq \tau \leq 0.15$*;* $u(\tau) = [4 \cdot m_5(\tau) - 1.74 \ \ m_2(\tau) - 1.74 \ \ 3 \cdot m_8(\tau) \ \ m_4(\tau) - 1.75]^T$ *for* $0.15 \leq \tau \leq 0.29$*;* $u(\tau) = [4 \cdot m_5(\tau) - 1.43 \ \ m_6(\tau) - 0.21 \ \ 3 \cdot m_8(\tau) \ \ m_4(\tau) - 1.22]^T$ *for* $0.3 \leq \tau \leq 0.52$*;* $u(\tau) = [4 \cdot m_5(\tau) - 3.04 \ \ m_6(\tau) - 2.04 \ \ 3 \cdot m_8(\tau) \ \ m_4(\tau) - 1]^T$ *for* $0.52 \leq \tau \leq 0.74$*. The trajectories for one and three intermediate states are illustrated in Fig. 4(a) and Fig. 4(b), respectively. In these figures dotted line shows the border between* $\mathcal{R}^1$ *and* $\mathcal{R}^2$*. The total time duration for several*



(a)



(b)

Figure 4: Trajectory for (a)1 int. state and (b)3 int. state for Ex. 7

intermediate states can be found in Table 1. The experiments were performed by a Matlab program running on a PC with Intel(R) Core(TM)2CPU T5600 @ 1.83GHz, 2.00 GB of RAM.

Table 1: Intermediate States

Number of int. states	0	1	3	9	13
Total duration (t.u.)	0.87	0.83	0.74	0.73	0.72
CPU time (sec.)	0.03	0.11	0.32	1.65	2.32

Example 8 *Let us consider the net in Fig. 5 with* $\boldsymbol{\lambda} = [3\ 1\ 1]^T$*. By using Algorithm 1, the total time to reach* $\boldsymbol{m}_f = [1\ 10\ 6\ 2]^T$ *from* $\boldsymbol{m}_0 = [13\ 3\ 1\ 10]^T$ *through a linear trajectory is obtained as* 2.43 *t.u., with the control* $u(\tau) = [3 \cdot m_4(\tau) - 7 \ \ m_2(\tau) \ \ m_3(\tau) - 1]^T$ *for* $0 \leq \tau \leq 0.93$*,* $u(\tau) = [3 \cdot m_4(\tau) - 12 \ \ m_4(\tau) \ \ m_3(\tau) - 1.71]^T$ *for* $0.93 \leq \tau \leq 1.26$*,* $u(\tau) = [3 \cdot m_1(\tau) - 3 \ \ m_2(\tau) \ \ m_3(\tau) - 0.42]^T$ *for* $1.26 \leq \tau \leq 2.43$*, while* $\boldsymbol{m}_f$ *was reached in* 4.4 *t.u. in ([10]). The piecewise linear trajectory with 4 intermediate states obtained by Algorithm 2 yields* 1.38 *t.u.*

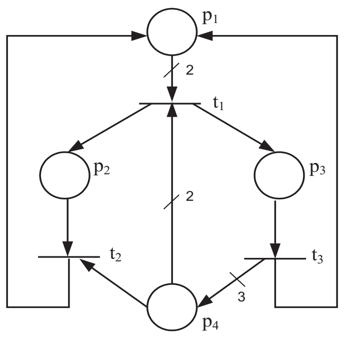


Figure 5: A PN

5 Closed Loop Control

Using Algorithm 2 we obtain an offline controller. Obviously, in a real system, perturbations affect the evolution and when the control law is implemented, the noise effect should be eliminated. In this section we assume that the PWL trajectory $path = \{\boldsymbol{m}_0,\ \boldsymbol{m'}^1, ..., \boldsymbol{m'}^s,\ \boldsymbol{m}_f\}$ has been computed using Alg. 2 and our problem is to follow this trajectory online. For the online implementation we consider the discrete-time representation of the system ([5]). Our approach will be: for each segment of trajectory we will apply during each discrete-step k the maximum possible flow solving a LPP. In fact, the procedure can be seen as a model predictive control scheme with timing horizon 1 and having as optimization index the minimization of the time. In ([5]) it is proved that using this control scheme the closed loop system is asymptotically stable.

The discrete-time representation of the continuous-time system (7) is given by:

$$\begin{array}{l} \boldsymbol{m}(k+1) = \boldsymbol{m}(k) + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}(k) \\ 0 \leq \boldsymbol{w}(k) \leq \boldsymbol{\Lambda} \cdot \boldsymbol{\Pi}(\boldsymbol{m}(k)) \cdot \boldsymbol{m}(k) \end{array} \tag{14}$$

Here Θ is the sampling period ($\tau = k \cdot \Theta$) and $\boldsymbol{m}(k)$ is the marking at step k, i.e., at time $k \cdot \Theta$. The sampling period should be small enough to avoid spurious states. For all p in P the following should be satisfied:

$$\sum_{t_j \in p^\bullet} \lambda_j \cdot \Theta < 1 \tag{15}$$

In order to design the closed loop control for *path*, we developed Alg. 3. Here s denotes the number of intermediate states. In the algorithm, the procedure is applied to each segment $\boldsymbol{m'}^j$ - $\boldsymbol{m'}^{j+1}$. Therefore, the first problem will be to reach $\boldsymbol{m'}^1$ from $\boldsymbol{m}_0$. In order to do this, LPP (16) computes at $k \cdot \Theta$ the maximum distance that can be executed from the actual marking, $\boldsymbol{m}(k)$, in the direction of $\boldsymbol{m'}^{j+1}$ during next Θ. Then, at next step, i.e., $(k+1)$, the new marking is obtained and LPP (16) is applied again. This LPP tries to maximize the distance during the next Θ that can be executed from the actual marking (constraint 1) such that the obtained intermediate marking $\boldsymbol{m'}_{next}$ belongs to the straight line from the actual marking $\boldsymbol{m}(k)$ to the final one $\boldsymbol{m'}^{j+1}$ (constraints 2 and 3) while the controlled flow is dynamically bounded (constraint 4). Then the same procedure is applied when the initial marking is $\boldsymbol{m'}^1$ and $\boldsymbol{m'}^2$ should be reached and so on. The procedure stops when $\boldsymbol{m}_f$ is reached. We will denote by $\boldsymbol{z}$ the noise that affect the flow (see eq. (17)).

Algorithm 3 Closed Loop Control

```
**Input:** $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $path$, $s$
$\boldsymbol{m'}^0 = \boldsymbol{m}_0$;  $\boldsymbol{m'}^{s+1} = \boldsymbol{m}_f$;
$\boldsymbol{m}(0) = \boldsymbol{m'}^0$;  $k = 0$;
**for** $j = 0 : s$ **do**
  **while** $||\boldsymbol{m}(k) - \boldsymbol{m'}^{j+1}|| > \epsilon$ **do**
    Solve the following LPP
```

$$\begin{array}{ll} \max & \alpha \\ \text{s.t.} & \boldsymbol{m'}_{next} = \boldsymbol{m}(k) + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w} \\ & \boldsymbol{m'}_{next} = (1-\alpha) \cdot \boldsymbol{m}(k) + \alpha \cdot \boldsymbol{m'}^{j+1} \\ & 0 \le \alpha \le 1 \\ & 0 \le \boldsymbol{w} \le \boldsymbol{\Lambda} \cdot \boldsymbol{\Pi}(\boldsymbol{m}(k)) \cdot min\{\boldsymbol{m}(k), \boldsymbol{m'}_{next}\} \end{array} \tag{16}$$

```
    Advance one step and obtain the new marking
```

$$\boldsymbol{m}(k+1) = \boldsymbol{m}(k) + \Theta \cdot \boldsymbol{C} \cdot (\boldsymbol{w} + \boldsymbol{z}) \tag{17}$$

```
    $k = k + 1$
  **end while**
**end for**
```

Some remarks regarding the online implementation:

- If there is no perturbation, total time to reach the desired marking should be, in general, smaller than the time obtained by the offline procedure. This is due to the fact that the flow in the offline computation that drive the system from a marking $\boldsymbol{m'}^j$ to a marking $\boldsymbol{m'}^{j+1}$ is constant and it is bounded by these two markings. In the online implementation, the flow at step k is bounded by the actual marking and $\boldsymbol{m'}^{j+1}$. Obviously, being the actual marking a convex combination of $\boldsymbol{m'}^j$ and $\boldsymbol{m'}^{j+1}$ the flow is greater, in general.

- Obviously, in some cases, the time obtained by the online computation can be greater. For example, let us assume that the optimal time computed offline to go from $\boldsymbol{m'}^j$ to $\boldsymbol{m'}^{j+1}$ is 0.3 t.u. Taking a sampling time equal to 0.25 and assuming a maximum flow during trajectory equal to 1, in the absence of any perturbation, at least two steps are required to reach the desired marking. This corresponds to, at least, 0.5 t.u. that is greater than the offline computation.

Example 9 *Let us consider the contPN system considered in Ex. 7 with the same initial and target marking:* $\boldsymbol{m}_0 = [7.5\ 4.5\ 4\ 2\ 1.5\ 5\ 4\ 2.5]^T$ *and* $\boldsymbol{m}_f = [7\ 5\ 5\ 1\ 1\ 4\ 5\ 3]^T$. *In Ex. 7, path was obtained as path* $= \{\boldsymbol{m}_0, \boldsymbol{m'}^1, \boldsymbol{m'}^2, \boldsymbol{m'}^3, \boldsymbol{m}_f\}$ *where* $\boldsymbol{m'}^1 = [7.5\ 4.5\ 4.26\ 1.74\ 1.5\ 4.74\ 4.26\ 2.76]^T$, $\boldsymbol{m'}^2 = [7.5\ 4.5\ 4.5\ 1.5\ 1.5\ 4.5\ 4.5\ 3]^T$ *and* $\boldsymbol{m'}^3 = [7.45\ 4.78\ 4.55\ 1.22\ 1.22\ 4.45\ 4.78\ 3]^T$.

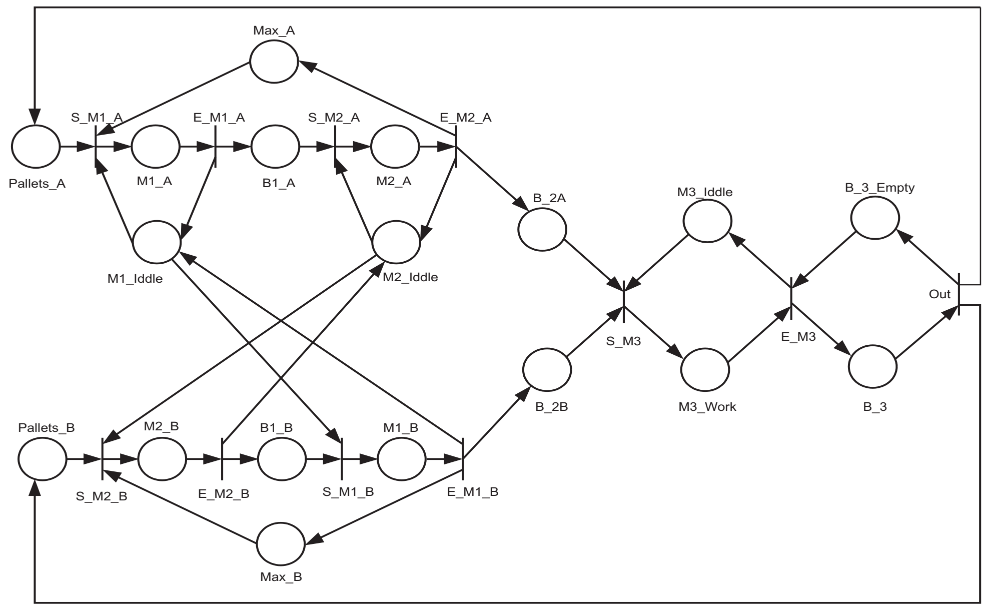


Figure 6: A contPN model of a flexible manufacturing system

According to inequality (15), $\Theta \leq 1/7$ *must be satisfied. Let us assume that* $\Theta = 0.01$ *and no perturbation. According to Alg. 3, we consider first the segment* $\boldsymbol{m}_0$ - $\boldsymbol{m'}^1$*. That is, LPP (16) is solved at* $k = 0$ *to calculate the maximum distance that can be executed from* $\boldsymbol{m}_0$ *during* Θ *t.u in the direction of* $\boldsymbol{m'}^1$*. This yields* $\boldsymbol{w}(1) = [2.02\ 2.02\ 0\ 2.02]^T$*. Applying the corresponding control to the system* $\boldsymbol{m}(1) = [7.5\ 4.5\ 4.02\ 1.98\ 1.5\ 4.98\ 4.02\ 2.52]^T$ *is reached.*

Since $||\boldsymbol{m}(1) - \boldsymbol{m'}^1|| > \epsilon$*, LPP(16) is solved considering* $\boldsymbol{m}(1)$ *the initial marking and the controlled flow is obtained equal to* $\boldsymbol{w}(2) = [1.99\ 1.99\ 0\ 1.99]^T$ *corresponding to* $\boldsymbol{m}(2) = [7.5\ 4.5\ 4.04\ 1.96\ 1.5\ 4.96\ 4.04\ 2.54]^T$*.*

Continuing applying Alg. 3, $\boldsymbol{m}_f$ *is reached from* $\boldsymbol{m}_0$ *after* 66 *discrete steps which correspond to* 0.66 *t.u.. This is smaller than the time obtained by the offline computation which was* 0.74 *t.u.*

6 Case study

In this section we will apply our heuristic method for minimum time control to a manufacturing system. The net in Fig.6 (considered also in [4]) represents a flexible manufacturing system. Let us assume the speed of all operations take 1 t.u., i.e., $\boldsymbol{\lambda} = \boldsymbol{1}$. A product is composed from two different parts, A and B. Parts A are processed in machine $M1$ and then in machine $M2$ while parts B are processed in machine M2 and after in machine $M1$. The intermediate products are stored in B_1A and B_1B, and the final products respectively in B_2A and B_2B. Machine $M3$ assembles part A and part B producing in this way the final product which is temporally stored in B_3. The parts are moved on pallets. Hence, in this system we have three sequential machines, 4 intermediate deposits and one terminal deposit.

Let us assume initially 35 pallets of type A ($\boldsymbol{m}_0[Pallets\_A] = 35$); 25 pallets of type B ($\boldsymbol{m}_0[Pallets\_B] = 25$); one machine of each type from which the first two are initially in *Idle* state while the third one is in *working* state ($\boldsymbol{m}_0[M1\_Iddle] = \boldsymbol{m}_0[M2\_Iddle] = \boldsymbol{m}_0[M3\_Work] = 1$); one final product of type A and one of type B produced ($\boldsymbol{m}_0[B\_2A] = \boldsymbol{m}_0[B\_2B] = 1$); a maximum of 10 parts of type A and B produced at each time ($\boldsymbol{m}_0[Max\_A] = \boldsymbol{m}_0[Max\_B] = 10$); and finally, the capacity of the buffer to store the final products just before the recycling of pallets equal to 25 ($\boldsymbol{m}_0[B\_3\_Empty] = 25$). Let us consider the task of producing 10 final product as fast as possible, in minimum time. Therefore, for the final marking we need to have: $\boldsymbol{m}_f[Pallets\_A] = 27$, $\boldsymbol{m}_f[Pallets\_B] = 17$, $\boldsymbol{m}_0[Max\_A] = \boldsymbol{m}_0[Max\_B] = 10$, $\boldsymbol{m}_0[M1\_Iddle] = \boldsymbol{m}_0[M2\_Iddle] = \boldsymbol{m}_0[M3\_Iddle] = 1$, $\boldsymbol{m}_0[B\_3\_Empty] = 15$ and $\boldsymbol{m}_0[B\_3] = 10$. Driving the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ through the linear trajectory assigning constant flows to transitions takes 1000 t.u.

In order to improve the time and refine the trajectory, we apply Algorithm 2. The resulting path is composed by 4 intermediate states. Therefore 5 line segments, i.e., $path = \{\boldsymbol{m}_0 \rightarrow \boldsymbol{m}'^1 \rightarrow \boldsymbol{m}'^2 \rightarrow \boldsymbol{m}'^3 \rightarrow \boldsymbol{m}'^4 \rightarrow \boldsymbol{m}_f\}$ (see Table 2) are obtained. The time spent through this PWL trajectory is 802 t.u. The controlled flows at each segment is given in Table 3.

Table 2: Intermediate States

place	$\boldsymbol{m}_0$	$\boldsymbol{m}'^1$	$\boldsymbol{m}'^2$	$\boldsymbol{m}'^3$	$\boldsymbol{m}'^4$	$\boldsymbol{m}_f$
Pallet_A	35	33.5	31	30	28.500	27
M1_A	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
B1_A	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
M2_A	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
MAX_A	10	10	10	10	10	10
M1_IDLE	1	1	1	1	1	1
M2_IDLE	1	1	1	1	1	1
Pallet_B	25	23.5	21	20	18.500	17
M2_B	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
B1_B	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
M1_B	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
Max_B	10	10	10	10	10	10
B2_A	1.01	1.01	0.51	0.015	0.01	ϵ
B2_B	1.01	1.01	0.51	0.015	0.01	ϵ
M3_Idle	ϵ	0.51	0.51	1	1.009	1.01
M3_Work	1.01	0.51	0.51	0.019	0.010	ϵ
B3_Empty	25	23	20	18.015	16.500	15
B_3	ϵ	2.01	5.01	6.995	8.509	10.01

Applying the closed loop control developed in Section 5 assuming $\Theta = 0.01$ which satisfies inequality (15) and zero noise effect, the total time to drive the system from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ through the path is obtained as 800 t.u. Under closed loop control, the evolution of markings of places B_2B and B_3; controlled flows of transitions S_M3, E_M3 are given in Fig. 7 and Fig. 8, respectively.

In order to see the noise effect on the closed loop controlled system, we

Table 3: controlled flows

transition	seg.1	seg.2	seg.3	seg.4	seg.5
S_M1_A	0.01	0.01	0.01	0.0099	0.01
E_M1_A	0.01	0.01	0.01	0.0099	0.01
S_M2_A	0.01	0.01	0.01	0.0099	0.01
E_M2_A	0.01	0.01	0.01	0.0099	0.01
S_M2_B	0.01	0.01	0.01	0.0099	0.01
E_M2_B	0.01	0.01	0.01	0.0099	0.01
S_M1_B	0.01	0.01	0.01	0.0099	0.01
E_M1_B	0.01	0.01	0.01	0.0099	0.01
S_M3	0.01	0.012	0.014951	0.01	0.01
E_M3	0.014	0.012	0.019852	0.01	0.01
Out	0	0	0	0	0

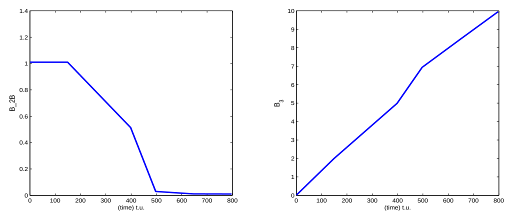


Figure 7: Evolution of m_{B_2B} and m_{B_3}

introduce noise on the flow of transitions E_M1_B, S_M3, E_M3 and Out during the time intervals [0 100] [250 350] [500 550]. The noise applied to transitions S_M3, E_M3, i.e. z_{S_M3}, z_{E_M3} are shown in Fig. 9.

Under this condition, when we apply developed closed loop control to the system, the total time is obtained as 940 t.u. The evolution of markings of places B_2B and B_3, and the controlled flows of transitions S_M3, E_M3 are given in Fig. 10 and Fig. 11, respectively.

7 Conclusions

The control problem adressed in the paper consists of reaching a target state in minimum time through a piecewise linear trajectory. Besides the piecewise linear dynamics of continuous Petri nets, the control method handles the fact that the input constraints depend on the current marking, i.e., the inputs are dynamically constrained.

The heuristics proposed in this paper computes first a "rough" piecewise linear trajectory that is refined afterwards in those intervals that allow an improvement. The heuristics makes use of BPPs to obtain intermediate states and LPPs to compute linear trajectories. The refinement of the trajectory is achieved recursively. Finally, a closed loop method is proposed similarly with a receding horizon scheme that ensures that the final marking is reached.

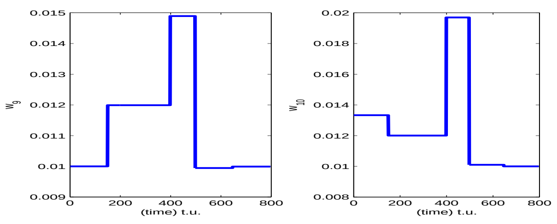


Figure 8: Evolution of w_{S_M3}, w_{E_M3}

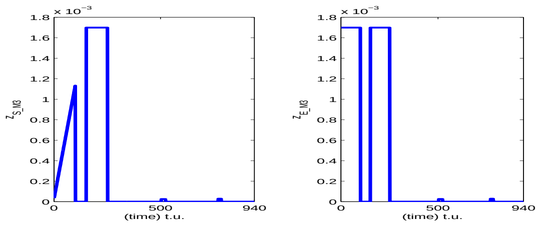


Figure 9: Noise effect on transitions z_{S_M3} and z_{E_M3}

References

[1] R. David and H. Alla. *Autonomous and timed continuous Petri nets.* Springer, Berlin, 2005.

[2] L.C.G.J.M. Habets, P.J. Collins, and J.H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *Automatic Control, IEEE Transactions on*, 51(6):938–948, June 2006.

[3] J. Júlvez, L. Recalde, and M. Silva. On reachability in autonomous continuous Petri net systems. In W. van der Aalst and E. Best, editors, 24^{th} *International Conference on Application and Theory of Petri Nets (ICATPN 2003)*, volume 2679 of *Lecture Notes in Computer Science*, pages 221–240, Eindhoven, The Netherlands, June 2003. Springer.

[4] J. Júlvez, L. Recalde, and M. Silva. Steady–state performance evaluation of continuous mono–t–semiflow petri nets. *Automatica*, 41(4):605–616, 2005.

[5] C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. Optimal model predictive control of timed continuous Petri nets. *IEEE Transactions on Automatic Control*, 53(7):1731 – 1735, August 2008.

[6] C. Mahulea, A. Ramirez, L. Recalde, and M.Silva. Steady state control reference and token conservation laws in continuous Petri net systems. *IEEE Transactions on Automation Science and Engineering*, 5(2):307–320, April 2008.

[7] C. Mahulea, L. Recalde, and M. Silva. Basic Server Semantics and Performance Monotonicity of Continuous Petri Nets. *Discrete Event Dynamic Systems: Theory and Applications*, 19(2):189 – 212, June 2009.

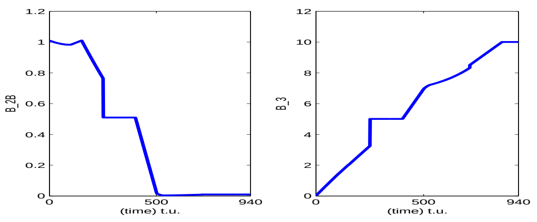


Figure 10: Evolution of m_{B_2B} and m_{B_3} under noise effect

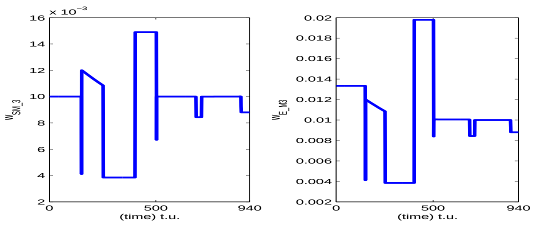


Figure 11: Evolution of w_{S_M3}, w_{E_M3} under noise effect

[8] V. F. Montagner, V.J.S. Leite, R.C.L.F Oliveira, and P.L.D. Peres. State feedback control of switched linear systems: An LMI approach. *Journal of Computational and Applied Mathematics*, 94(2):192–206, 2006.

[9] M. Silva and L. Recalde. On fluidification of Petri net models: from discrete to hybrid and continuous models. *Annual Reviews in Control*, 28(2):253–266, 2004.

[10] J. Xu, L. Recalde, and M. Silva. Tracking control of timed continuous Petri net systems under infinite servers semantics. In *17th World Congress of IFAC*, pages 3192–3197, Seoul, Korea, 2008.

[11] H. Yang, G. Xie, T. Chu, and L. Wang. Commuting and stable feedback design for switched linear systems. *Journal of Computational and Applied Mathematics*, 94(2):192–206, 2006.

[12] M.C. Zhou, F. DiCesare, and D. Guo. Modeling and performance analysis of a resource-sharing manufacturing system using stochastic Petri nets. In *Fifth IEEE Symp. on Intelligent Control*, pages 1005–1010, Phildephia, PA, 1990.