Target Marking Calculation for Timed Continuous Petri Nets

Hanife Apaydin Özkan
Department of Electrical and Electronics Engineering
Anadolu University
Eskişehir, Turkey
hapaydin1@anadolu.edu.tr

Abstract—Continuous Petri nets were introduced as a fluid approximation of discrete Petri nets which may suffer from state explosion problem. In this paper, timed continuous Petri net systems which are piecewise linear systems with input constraints are considered. Particularly, the control problem in which some components of target marking are not specified is concentrated. Unspecified components of the target marking are calculated on the basis of a nonlinear programming that minimizes time to drive the system from initial marking through a linear trajectory. An online control algorithm is used to drive the system from given initial marking to the completely determined target marking.

Keywords—Continuous Petri net, optimization, control

I. INTRODUCTION

Petri nets (PNs) is a well known formalism used for modeling, analysis and synthesis of Discrete Event Systems (DESs). Like other modeling formalisms for DESs, PNs suffer from the so called state explosion which leads to an exponential growth of the size of state space with respect to the size of the system and population of initial state. One way to deal with that problem is to use some kind of relaxations [1,2,3,4]. Fluidification is a classical relaxation technique, and may be very useful when applied to highly populated systems.

For PNs, fluidification was introduced in [4,5] aiming at giving fluid (continuous) approximation of original PN in the sense of behaviours and properties, and these models are called continuous Petri nets. The idea is to try to overcome, at least partially, the potentially very high computational complexity arising in many practical situations.

Different techniques have been proposed for control of continuous Petri nets in the literature [6,7,8-10]. Steady state optimal control of continuous Petri nets was studied in [11] where it was shown that, the optimal steady state control problem of a continuous Petri net system can be solved by means of Linear Programming Problem (LPP) in the case that all transitions are controllable and the objective function is linear. The transitory control problem is also solved by means of implicit and explicit Model Predictive Control (MPC) strategy [7]. The step tracking problem, i.e. design of control laws to drive the system states to target references was considered in [10] and a Lyapunov-function-based dynamic control algorithm was proposed for the problem. That method requires solving a BiLinear Programming Problem (BLP) for the computation of intermediate states. In [6], an efficient heuristics for minimum time control of continuous Petri nets, which aims at driving the system from an initial state to a target one through a piecewise linear trajectory is developed.

In some control problems, final states of some places may not be specified, because of some requirements other than a specific final state for these places, such as minimizing/maximizing the marking of these places, reaching desired final states in other places in minimum time and etc. For example, in a home appliance plant final number of raw products in buffers may not be important while producing specific number of final products in storehouse in minimum time is significant. As another example, due to a customer order, a factory which produces A and B products may have to produce required number of A in minimum time, meanwhile the number of B produced is not mandatory. Accordingly, corresponding components of the target state are not specified in the control problem. In the author’s previous work, [12], a final marking planning method is introduced for timed continuous Petri nets for the first time in the literature. But in that paper, only a subclass of timed continuous Petri nets and only time minimization objective were under consideration. Now, we extend that work for general timed continuous Petri nets and illustrate the calculation of target marking under different objective functions. Unspecified components of the target marking are calculated on the basis of a nonlinear programming techniques. A simplified form of the online control algorithm developed in [6] is used to drive the system from the given initial marking to the completely determined target marking through a linear trajectory.

The remainder of the paper is organized as follows. Section II briefly introduces the required concepts of
continuous Petri net systems, while Section III introduces the formulation of applied control. In Section IV, the proposed method for calculating unspecified components of target marking under time minimization objective is addressed. Online implementation of the controlled system's evaluation is represented in Section V. A table factory system is taken as a case study to illustrate the procedure in Section VI. Finally, Section VII summarizes the main conclusions of the work.

II. CONTINUOUS PETRI NETS

This section introduces the main concepts related to continuous Petri nets and timed continuous Petri nets. The reader is assumed to be familiar with basic PN concepts (see [13,14]).

A. Untimed Continuous Petri nets

**Definition 2.1** A continuous Petri net system is a pair \( \langle N, m_0 \rangle \), where \( N = \langle P, T, \text{Pre}, \text{Post} \rangle \) is the net structure where \( P \) and \( T \) are the sets of places and transitions respectively; \( \text{Pre}, \text{Post} \subset \mathbb{N}^{\mid P \mid \times \mid T \mid} \) are the pre and post matrices; \( m_0 \in \mathbb{R}_{\geq 0}^{|P|} \) is the initial marking (state).

For \( v \in P \cup T \), the sets of input and output nodes are denoted as \( *v \) and \( v^* \), respectively. Let \( p_i, i \in \{1, 2, ..., |P|\} \) and \( t_j, j \in \{1, 2, ..., |T|\} \) denote place and transition, respectively. \( \text{Pre}(p_i, t_j) \) or \( \text{Pre}_v \) denotes the weight of arc directed from place \( p_i \in P \) to transition \( t_j \in T \) and \( \text{Post}(p_i, t_j) \) or \( \text{Post}_v \) denotes the weight of arc directed from transition \( t_j \in T \) to \( p_i \in P \).

\( m \in \mathbb{R}_{\geq 0} \) is the marking vector and represents the distribution of tokens in places. Each place can contain a nonnegative real number of tokens and \( m(p_i) \) indicates the number of tokens in place \( p_i \in P \). A transition \( t_j \in T \) is enabled at \( m \) iff \( \forall p_i \in *t_j, m_i \geq 0 \) and its enabling degree is given by

\[
\text{enab}(t_j, m) = \min_{p_i \in t_j} \left\{ \frac{m(p_i)}{\text{Pre}(p_i, t_j)} \right\}
\]

which represents the maximum amount in which \( t_j \) can fire. An enabled transition \( t_j \) can fire in any real amount \( \alpha \), with \( 0 < \alpha \leq \text{enab}(t_j, m) \) leading to a new state \( m' = m + \alpha C(., t_j) \) where is the token flow matrix and \( C = \text{Post} - \text{Pre} \) is its \( j^{th} \) column.

If \( m \) is reachable from \( m_0 \) through a finite sequence \( \sigma \) the state (or fundamental) equation is satisfied:

\[
m = m_0 + C\sigma
\]

where \( \sigma \in \mathbb{R}^\|\sigma\|_\infty \) is the firing count, i.e., \( \sigma_j \) is the cumulative amount of firings of \( t_j \) in the sequence \( \sigma \).

B. Timed Continuous Petri nets

**Definition 2.2A** A timed continuous Petri net (contPN) system \( \langle N, A, m_0 \rangle \) is a continuous Petri net system together with a vector \( A \in \mathbb{R}^\|A\|_\infty \) where \( \lambda_j \) is the firing rate of \( t_j \).

As in untimed continuous Petri nets, state equation summarizes the way the marking evolves along time. The state equation of a contPN has an explicit dependence on time \( m[\tau] = m_0 + C\sigma[\tau] \) where \( \tau \) is global time.

But, in continuous systems, the marking is continuously changing, so we may consider the derivative of \( m \) with respect to time. By this way \( m[\tau] = \dot{C}\sigma[\tau] \) is obtained. Here, \( \dot{\sigma} \) is flow through transitions and it is denoted by \( f[\tau] = \dot{\sigma}[\tau] \). Hence, the state equation is

\[
m[\tau] = \dot{C}.f[\tau]
\]

For the sake of simplicity \( \tau \) is omitted in the rest of the paper. Different semantics have been defined for continuous timed transitions [4,15]. This paper is focused on infinite server semantics. Under this semantics, \( f = A \Pi[m]m \).
where $A = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is the firing rate matrix and $\Pi[m]$ is the constraint matrix at marking $m$ defined by elements:

$$
\Pi[m]_{ij} = \begin{cases} 
\frac{1}{\text{Pre}_{ij}}, & \text{if } \frac{m_i}{\text{Pre}_{ij}} = \min_{p_i \in \text{set}_i} \left\{ \frac{m_i}{\text{Pre}_{ij}} \right\} \\
0, & \text{otherwise}
\end{cases}
$$

(5)

Note that the value of $\Pi[m]$ changes when the system switches its configuration: a configuration assigns to each transition one place that will control its firing rate (i.e., it is constraining that transition). The number of configurations is upper bounded by

$$
\gamma = \prod_{j=1}^{T} |I_j|
$$

The state space (or reachability set) of a contPN system can be divided into regions according to configurations as: $\mathcal{R} = \mathcal{R}^1 \cup \ldots \cup \mathcal{R}^\gamma$. Intuitively, each $\mathcal{R}^z$ denotes a region where the flow is limited by the same subset of places (one for each transition). If a marking $m$ belongs to $\mathcal{R}^z$, we denote $\Pi[m] = \Pi^z$ the corresponding constraint matrix (if $m$ is a border point and belongs to several regions, $\Pi[m]$ is the constraint matrix of any of these regions).

**Example 1:** Consider the contPN in Figure 1. Assume $\lambda_1 = \lambda_2 = \lambda_3 = 1$. The system dynamics is described as follows:

$$
\begin{align*}
\dot{m}_1 &= -2f_1 + f_2 + f_3 = -\min\{m_1, m_2\} - \min\{m_2, m_4\} + m_1 \\
\dot{m}_2 &= f_1 - f_2 = \min\{m_1, m_4\} - \min\{m_2, m_4\} \\
\dot{m}_3 &= f_1 - f_3 = \min\{m_1, m_4\} - m_3 \\
\dot{m}_4 &= -2f_1 - f_2 + 3f_3 = -\min\{m_1, m_4\} - \min\{m_2, m_4\} + 3m_4
\end{align*}
$$

(6)

![Figure 1. A contPN](image)

III. PROBLEM FORMULATION

In this work, we assume that the only action that can be applied to a contPN is to reduce the flow of transitions [16]. If a transition can be controlled (its flow can be reduced or even stopped), we will say that it is a controllable transition [11]. In this work, it is assumed that all transitions are controllable.

**Definition 3.1** The controlled flow, $w$, of a contPN is defined as $w[\tau] = f[\tau] - u[\tau]$ with $0 \leq u[\tau] \leq f[\tau]$, where $f$ is the flow of the uncontrolled system, i.e., defined as in (4), and $u$ is the control action.

Therefore, the control input $u$ is dynamically upper bounded by the flow $f$ 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:

---

1These regions are disjoint except possibly on the borders.
\[\dot{m} = C \cdot (f - m) = C \cdot w \quad (a)\]
\[0 \leq w \leq A \Pi[m] \cdot m \quad (b)\]

In this work, we focus on control of contPNs in the case that some components of target markings are not specified. Section IV proposes a method to compute the unspecified components of target markings which are not specified in advance. Then in Section V, an online control algorithm is used to drive the system from its initial marking, \(m_0\), to the computed target marking, \(m_f\), through a linear trajectory.

We assume that \(m_0\) and \(m_f\) are strictly positive. The assumption that \(m_0\) is positive ensures that the system can move at \(\tau = 0\) in the direction of \(m_f\) \([17]\); the assumption that \(m_f\) is positive ensures that \(m_f\) can be reached in finite time \([11]\).

### IV. COMPUTATION OF UNSPECIFIED COMPONENTS OF TARGET MARKING

In a contPN, it may occur that some components are intended to move to a specific target while the other components are not definite. These indefinite markings should be resolved by taking profit goals and priorities into account. In control systems, one of the most common control objectives is time minimization. Hence, we will focus on this objective. However, the objective function might be changed according to user demands. This case is illustrated in the case study.

In this work, we aim specified components to move towards the desired target marking in minimum time through a linear trajectory. Since it is assumed that \(0 \cdot 0 \cdot m_0 \cdot m_f \cdot \alpha \in [01]\) then the linear trajectory from \(m_0\) to \(m_f\) can be followed by the system \([17]\). At each marking \(m\) on the line connecting \(m_0\) to \(m_f\) (i.e., \(m = \alpha \cdot m_0 + (1 - \alpha) \cdot m_f, \quad \alpha \in [01]\) the controlled flow in the direction to \(m_f\) and minimizing the reaching time can be calculated by

\[w = \alpha \cdot w_0 + (1 - \alpha) \cdot w_f, \quad \alpha \in [01]\]  

where \(w_0\) denotes the maximum flow at the initial marking in the direction from \(m_0\) to \(m_f\), and \(w_f\) denotes the maximum flow at the final marking in the direction from \(m_0\) to \(m_f\).

**Proposition 1:** Let \(\langle N, m_0 \rangle\) be a contPN. At each marking \(m\) on the line connecting \(m_0 \in \mathbb{R}_{\geq 0}\) to \(m_f \in \mathbb{R}_{\geq 0}\) (\(m = \alpha \cdot m_0 + (1 - \alpha) \cdot m_f, \quad \alpha \in [01]\)), the controlled flow \(w = \alpha \cdot w_0 + (1 - \alpha) \cdot w_f, \quad \alpha \in [01]\)

a) drives the system from \(m_0\) to \(m_f\),

b) satisfies \(0 \leq w \leq f\) which guarantees \(0 \leq u \leq f\) at that point.

**Proof 1:**

a) \(w_0\) and \(w_f\) satisfy the following:

\[m_f = m_0 + C \cdot w_0 \cdot \tau_0\]
\[m_f = m_0 + C \cdot w_f \cdot \tau_f\]  

where \(\tau_0\) and \(\tau_f\) are the times required to reach \(m_f\) from \(m_0\) by \(w_0\) and \(w_f\), respectively. Let \(s = w_0 \cdot \tau_0 = w_f \cdot \tau_f\), then

\[m_f = m_0 + C \cdot s\]  

If we apply \(w = \alpha \cdot w_0 + (1 - \alpha) \cdot w_f\) at a marking \(m = \alpha \cdot m_0 + (1 - \alpha) \cdot m_f, \quad (\alpha \in [01])\), that yields

\[m_f = \alpha \cdot m_0 + (1 - \alpha) \cdot m_f + C \cdot (\alpha \cdot w_0 + (1 - \alpha) \cdot w_f \cdot \tau)\]

By using equations (10) and (11), \(s = (\alpha \cdot w_0 + (1 - \alpha) \cdot w_f \cdot \tau) / \alpha \) is obtained. Since that equation can be solved for \(\tau\) at any \(\alpha \in [01]\) (at any marking \(m = \alpha \cdot m_0 + (1 - \alpha) \cdot m_f, \quad \alpha \in [01]\)) can be driven to \(m_f\) by applying \(w = \alpha \cdot w_0 + (1 - \alpha) \cdot w_f\).

b) The controlled flows at \(m_0\) and \(m_f\) are calculated by \(w_0 = A \Pi[m_0] \cdot m_0, w_f = A \Pi[m_f] \cdot m_f\) respectively. According to (7)(b), \(0 \leq w_0 \leq A \Pi[m_0] \cdot m_0\) and \(0 \leq w_f \leq A \Pi[m_f] \cdot m_f\) must be satisfied.
Due to (5), $H_a$ is the constraint matrix of $m_a \in \mathcal{R}$ (i.e. $H_a = \Pi[a]$) if $H_a m_a = \min_{m \in \mathcal{R}} \{\Pi[m] m_a\}$. Hence

\begin{align*}
0 \leq w_0 & \leq A \Pi[m_0] m_0 \leq A \Pi[m] m_0 \quad m \in \mathcal{R} \\
0 \leq w_f & \leq A \Pi[m_f] m_f \leq A \Pi[m] m_f \quad m \in \mathcal{R}
\end{align*}

(12)

At each marking $m = \alpha_k m_0 + (1-\alpha_k) m_f$, $\alpha_k \in [0,1]$, the uncontrolled flow is calculated by $f = A \Pi[m] m$ which can be rewritten as

\[ f = A \Pi[m] \alpha_k m_0 + A \Pi[m] (1-\alpha_k) m_f, \quad \alpha_k \in [0,1] \]

(13)

According to (12) and (13), following inequality can be written

\[ 0 \leq w = \alpha_k w_0 + (1-\alpha_k) w_f \leq f = A \Pi[m] (\alpha_k m_0 + (1-\alpha_k) m_f), \quad \alpha_k \in [0,1] \]

(14)

That is, $\theta \leq w \leq f$ which guarantees that $\theta \leq u \leq f$.

Our method consists of calculating unspecified components of the target marking by maximizing the flows $w$ and $f$ at each point on the trajectory.

Maximum flow of transitions at $m_0$ in the direction to $m_f$ is calculated by minimizing the corresponding time $\tau_0$ with the following LPP, where $s_0 = w_0 \cdot \tau_0$

\[ \min_{s_0} \quad \tau_0 \]

s.t. $m_f = m_0 + C \cdot s_0$ (a)

$0 \leq s_0 \leq A \Pi[m_0] m_0 \cdot \tau_0$ (b)

(15)

The equations correspond to: (a) the straight line connecting $m_0$ to $m_f$, (b) flow constraints at $m_0$.

Maximum flow of transitions at $m_f$ in the direction to $m_f$ is calculated by minimizing the corresponding time $\tau_f$ with the following LPP, where $s_f = w_f \cdot \tau_f$

\[ \min_{s_f} \quad \tau_f \]

s.t. $m_f = m_0 + C \cdot s_f$ (a)

$0 \leq s_f \leq A \Pi[m_f] m_f \cdot \tau_f$ (b)

(16)

The equations correspond to: (a) the straight line connecting $m_0$ to $m_f$, (b) flow constraints at $m_f$.

**Proposition 2:** Let $\langle N, A, m_0 \rangle$ be a contPN system with $m_0 > 0$. If $m_f$ belongs to $\mathcal{R}$, then LPPs in (15) and (16) are feasible.

**Proof 2:** Since $m_f$ is a reachable marking, then there exists $s$ such that the state equations (15)(a) and (16)(a) are satisfied. By taking $\tau_0$ and $\tau_f$ sufficiently large (15)(b) and (16)(b) can be satisfied since $\lambda_f \Pi[m_f]/m_{0f} > 0$ and $\lambda_f \Pi[m_f]/m_{ff} > 0$.

In this work, the set of places whose final markings are specified precisely is denoted by $P_{sp} = \{p_i \mid m_f i \text{ is specified}\}$ and the set of places whose final markings are not specified is denoted by $P_{un} = \{p_i \mid m_f i \text{ is un-specified}\}$. In order to calculate $m_f \forall p_i \in P_{un}$ by minimizing the time, we propose to solve programming problem in (17) which is obtained by combining LPP (15) and LPP (16). In (17) $m_f \forall p_i \in P_{un}$ is calculated with variables $\tau_0, \tau_f, s_0, s_f, m_f \forall p_i \in P_{un}$ where $s_0 = w_0 \cdot \tau_0$, $s_f = w_f \cdot \tau_f$

\[ \min_{\tau_0, \tau_f, s_0, s_f, m_f \forall p_i \in P_{un}} \tau_0 + \tau_f \]

s.t. $m_f = m_0 + C \cdot s_0$ (a1)

$0 \leq s_0 \leq A \Pi[m_0] m_0 \cdot \tau_0$ (b1)

$m_f = m_0 + C \cdot s_f$ (a2)

$0 \leq s_f \leq A \Pi[m_f] m_f \cdot \tau_f$ (b2)

(17)
V. ONLINE IMPLEMENTATION OF CONTROL

For the online implementation, we will use a simplified form of the algorithm developed in [6] (see Algorithm 1). This algorithm considers the discrete-time representation [7] of the system and applies the maximum possible flow obtained by solving an LPP at each discrete-step \( k \) through the trajectory. In [7], it was proved that the closed loop system is asymptotically stable using this control scheme.

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

\[
m[k+1] = m[k] + \Theta C \cdot w[k] \quad (a)
\]

\[
0 \leq w[k] \leq |A| \cdot |m[k]| \cdot m[k] \quad (b)
\]

Here \( \Theta \) is the sampling period \( \tau = k \cdot \Theta \) and \( m[k] \) is the marking at step \( k \), i.e., at time \( \tau = k \cdot \Theta \). The sampling period should be small enough to avoid spurious states. For all \( p \in P \), the following should be satisfied [8]:

\[
\sum_{t \in p} \lambda_t \cdot \Theta < 1 
\]

LPP (20) computes the maximum distance that can be executed from the actual marking, \( m[k] \), in the direction of \( m_f \) during the next \( \Theta \) t.u. at \( k \cdot \Theta \). This LPP tries to maximize the distance during the next \( \Theta \) that can be executed from the actual marking (constraint 1) such that the obtained intermediate marking \( m' \) belongs to the straight line from the actual marking \( m[k] \) to the final one \( m_f \) (constraints 2 and 3) while the controlled flow is dynamically bounded (constraint 4).

Algorithm 1 implements the system evolution by calculating the controlled flows (hence, the required control action) and markings at each time step.

**Algorithm 1:** Online Closed Loop Control

**Input:** \( \langle N, m_0 \rangle, m_f \)

\( m[0] = 0; \quad k = 0; \)

**WHILE** \( || m[k] - m_f || > \epsilon \)

Solve the following LPP

\[
\begin{align*}
\text{max} & \quad \alpha \\
\text{s.t.} & \quad m' = m[k] + \Theta C \cdot w \\
& \quad m' = (1 - \alpha) m[k] + \alpha m_f \\
& \quad 0 \leq \alpha \leq 1 \\
& \quad 0 \leq w \leq |A| \cdot |m[k]| \cdot \min \{m[k], m'\}
\end{align*}
\]

Advance one step and obtain the new marking:

\[
m[k+1] = m[k] + \Theta C \cdot w
\]

\( k = k + 1 \)

**End WHILE**

**Example 2:** Let us consider the contPN system in Figure 1 with \( \lambda_1 = \lambda_2 = \lambda_3 = 1 \). Assume \( m_0 = [3 \ 3 \ 2 \ 2]^T \), \( P_{sp} = \{ p_1 \} \) and \( P_{sm} = \{ p_2, p_3, p_4 \} \). Due to the given sets of \( P_{sp} \) and \( P_{sm} \), the final marking of \( p_1 \) must be strictly specified, while that of others are not. Let \( m_{f_1} = 4.5 \).

By solving the programming problem in (17), markings of the unspecified places are obtained as \( m_{f_2} = 2.25, m_{f_3} = 1.25 \) and \( m_{f_4} = 3.5 \), that is \( m_f = [4.5 \ 2.25 \ 1.25 \ 3.5]^T \). By executing Algorithm 1 with \( k = 0.01 \), \( m_f \) is reached by 470 discrete steps (0.47 t.u.).
We use Algorithm 1 for the online implementation. Placement of $m_0$ and $m_f$ on $m_2 - m_4$ plane, marking and flow evolution of the system with respect to time are given in Figures 2 and 3, respectively.

**Figure 2:** Placement of $m_0$ and $m_f$ on $m_2 - m_4$ plane

**Figure 3:** Evolution of markings and controlled flows for Example 2
VI CASE STUDY

Let us consider the contPN sketched in Figure 4 (taken from [18]) which models a table factory system. This system consists of two different machines to make table-legs ($t_1$ and $t_2$), a machine to produce the table boards ($t_3$), a machine to assemble four legs and a board ($t_4$), a big painting line which paints two tables at once ($t_6$). More unpainted tables are sent ($t_7$) from another factory. The places $p_1$, $p_2$ and $p_3$ are work orders; while $p_4$, $p_5$ and $p_7$ are devoted to the storage of table-legs, boards and unpainted tables, respectively (see [18] and [19] for details).

Suppose in the initial marking $m_0 = [2 2 1 3 1 1]^T$, i.e. $m_0 = [2 2 1 3 1 1]^T$. For the final state we want to double unpainted tables in buffer $p_7$ ($m_f = 2$), while the other buffers ($p_5$ and $p_6$) keep the same number of products ($m_{i_5} = m_{i_6} = 1$).

By solving (17), the final marking satisfying required markings of places $p_3$, $p_5$, and $p_7$ and maximizing the flows at $m_0$ and $m_f$ directed from $m_0$ to $m_f$ is obtained as $m_f = [2 2 1 2 1 1 2]^T$. Algorithm 1 is applied for the implementation of the system evolution. System reaches the target marking in 0.4 t.u. Marking and flow evolution of the system with respect to time are given in Figure 5.

![Figure 4: A contPN model of a table factory](image_url)

![Figure 5: Evolution of markings and controlled flows for the case study with $m_0 = [2 2 1 3 1 1]^T$](image_url)
Let us consider a different initial state $m_0 = [3 \ 2 \ 1 \ 2 \ 1 \ 1 \ 3]^T$. For the final state, we want to decrease the working order of table leg machine $m_{f_3}$, i.e. $m_{f_3} = 1.5$, and keep the initial value of the table leg machines $m_{f_1}$, $m_{f_3}$, i.e. $m_{f_1} = 2$ and $m_{f_3} = 1$. Moreover, instead of time minimization, we want to minimize the number of unpainted tables in buffer $p_i$. Hence, the objective function should minimize $m_{f_3}$ instead of time minimization. For this, objective function of programming problem, (17) is changed as follows:

$$\min_{m_0, m_1, m_2, m_3, m_4, \forall p_i \in P_i} \left( m_{f_3} \right)$$  \hspace{1cm} (21)

Target marking is obtained as $m_f = [1.5 \ 2 \ 1 \ 3.1 \ 6.9 \ 2.1 \ 0.8]^T$. Algorithm 1 is applied for the implementation of the system evolution. System reaches the target marking in 2.2 t.u.

Marking and flow evolution of the system with respect to time are given in Figure 6.

![Figure 6](image_url)

**Figure 6**: Evolution of markings and controlled flows for the case study with $m_0 = [3 \ 2 \ 1 \ 2 \ 1 \ 1 \ 3]^T$

**VII CONCLUSION**

This paper is focused on contPNs which is a fluid approximation of discrete PNs in the sense of behaviours and properties. As differ from conventional Petri nets, a transition in a contPN can be fired in a real quantity instead of nonnegative integers. The control problem of contPN in which some components of target marking are not specified is concentrated. Unspecified components of target marking are calculated on the basis of a nonlinear programming that minimizes time to drive the system from the given initial marking. Moreover, choosing other objective functions for target marking calculation is also considered. After all components of target marking are completely specified, an online control algorithm is used to drive the system from the initial state to the completely specified target state through a linear trajectory. For the case study, a table factory system is studied.

**REFERENCES**


