Trabajo Fin de Máster

Máster en Ingeniería de Sistemas e Informática

Minimum-time Control for Structurally Persistent Continuous Petri Nets and The Application in Distributed Control

Liewei WANG

Director: Cristian Mahulea, Jorge Júlvez, and Manuel Silva

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
Universidad de Zaragoza

Septiembre 2010

Abstract

In this report, we first address the minimum-time control problem of structurally persistent timed continuous Petri Net systems (*ContPN*). In particular, an *ON-OFF* controller is proposed to drive the system from a given initial marking to the final marking in minimum-time. The controller is developed first for the discrete-time system ensuring that all transitions are fired as fast as possible in each sampling period until the required total firing counts are reached. After that, they are stopped suddenly. By taking the limit of the sampling period, the controller for continuous-time systems is obtained. Simplicity and the fact that it ensures minimum-time are the main advantages of the controller. A manufacturing system is taken as case study to illustrate the control strategy.

In a distributed controlled system, normally a complex dynamic system, the controllers are not centralized in one location, but are distributed in subsystems. We try to apply the *ON-OFF* controller into the distributed control of large scale systems modeled with timed continuous Petri net. The original net system is first structurally decompose into smaller subnets through sets of places. Then the *ON-OFF* controller is applied in controlling each subsystem. Algorithms are proposed to compute admissible control laws for the local subsystems in a distributed way. It is proved that with that control laws, the final state can be reached in minimum time.

Keyword: Petri nets, minimum-time control, distributed control

Contents

Chapter 1

Introduction

Petri Nets (PN) is a well known paradigm used for modeling, analysis, and synthesis of *discrete event systems* (DES). With strong facility to depict the sequence, concurrency, conflict and other synchronous relationships, it is widely applied in the industry, for the analysis of manufacturing, traffic, and software systems, etc. Similar to other modeling formalisms for DES, it also suffers from the *state explosion* problem. To overcome it, a classical relaxation technique called *fluidification* can be used.

Continuous Petri nets [7, 19] are fluid approximations of classical *discrete Petri nets* obtained by removing the integrality constrains, which means the firing counter vector and consequently the marking is no longer restricted to be in the naturals but relaxed into the non-negative real numbers. An important advantage of this relaxation is that more efficient algorithms for their analysis, e.g., reachability and controllability [14, 10] problems can be used.

Different approaches have been proposed in the literature for the control of different classes of $ContPNs$, e.g., First-Order Hybrid Petri nets [5] or finite server semantics [2] etc. In this work, the minimum-time control problem of *timed continuous Petri nets* under *infinite server semantics* is considered. For this class of systems several control approaches have been considered. In [14], the steady state control and optimal steady state control are studied. Model Predictive Control (MPC) is used for optimal control problem in [13] assuming a discrete-time model. In [24], a Lyapunov-function-based dynamic control algorithm is studied while in [3] an efficient heuristics for minimum-time control is proposed.

Here, we design an *ON-OFF* controller for structurally persistent $ContPN$ systems. With this controller, we will prove that the system is driven from an initial marking to a final one in minimum-time. The basic idea of the proposed control strategy is to fire every transition as fast as possible until the required total firing count is achieved (ON), and then it is stopped (OFF). This kind of controller has been studied in the case of linear systems [16, 21] and it is proved to be minimum-time in some cases. Unfortunately our system is only piecewise linear and the classical results can not be applied.

In the context of distributed systems, distributed timed automata is discussed in [11]. In [25], the author proposed a method for the modeling and decomposition of the large and complex discrete event manufacturing systems, where a Petri net based controller is distributed in machines and exchanges signals with coordinators. An architecture for distributed implementation of Petri nets in control applications is proposed in [17]. A distributed control strategy is designed in [8] for forbidden state avoidance for discrete event systems which are modeled as Petri nets.

Coming back to the continuous Petri nets, in [4], a reachability control problem of timed distributed continuous Petri net system is considered, which is composed of several subsystems that communicate through channels modeled by places. The proposed algorithm allows the subsystems to reach their respective target markings at different time instants and keep them as long as required.

In this work, the distributed control of large scale systems which are modeled with timed continuous Petri nets is addressed. As a starting point of this research topic, it is assumed the systems we handle are modeled with marked graphs. The idea is first structurally decomposing a large scale system into smaller ones, then applying the *ON-OFF* controller to each subsystem. Algorithms are proposed to computer the local control law separately. It is proved that with these control laws all the local controllers work independently, and the final state can be reached in minimum time.

This report is organized as follows: Chapter 2 briefly recalls some basic concepts on $ContPN$ and implicit places which will be used in the approximation of systems. In Chapter 3, an *ON-OFF* controller is proposed to drive the structurally persistent Petri net system from the initial state to final state in minimum-time. Chapter 4 tries to apply the *ON-OFF* controller to the distributed control of large scale systems modeled with $ContPN$ marked graphs. Some conclusions can be found in Chapter 5

Chapter 2

Basic concepts

The reader is assumed to be familiar with basic Petri net concepts (see [7, 19] for a gentle introduction).

2.1 Continuous Petri Nets

Definition 2.1.1. *A continuous Petri net system is a pair* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *where* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *is a net structure where:*

- *P and T are the sets of places and transitions respectively.*
- *$\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{R}_{\geq 0}^{|P| \times |T|}$ are the pre and post matrices.*
- *$\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$ is the initial marking (state).*

For $v \in P \cup T$, the sets of its input and output nodes are denoted as ${}^\bullet v$ and $v^\bullet$, respectively. Let p_i, $i = 1, \ldots, |P|$ and t_j, $j = 1, \ldots, |T|$ denote the places and transitions. Each place can contain a non-negative real number of tokens, this number represents the marking of the place. The distribution of tokens in places is denoted by $\boldsymbol{m}$. A transition $t_j \in T$ is enabled at $\boldsymbol{m}$ iff $\forall p_i \in^\bullet t_j$, $m(p_i) > 0$ and its enabling degree is given by

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

which represents the maximum amount in which t_j can fire. Transition t_j is called *k-enabled* under marking $\boldsymbol{m}$, if $enab(t, \boldsymbol{m}) = k$. 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 $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}(\cdot, t_j)$ 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: $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}$, where $\vec{\sigma} \in \mathbb{R}_{\geq 0}^{|T|}$ is the *firing count vector*, i.e., $\vec{\sigma}(t_j)$ is the cumulative amount of firings of t_j in the sequence σ.

If for all $p \in P$, $|p^\bullet| = 1$ then $\mathcal{N}$ is called *structurally persistent PN*, in the sense that independently by the initial marking, the net has no conflict.

Property 2.1.1. *[12] Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a structurally persistent PN system. If t_j is k-enabled at $\boldsymbol{m}$, it will remain k-enabled until t_j is fired.*

In timed continuous Petri net($ContPN$) the state equation has an explicit dependence on time: $\boldsymbol{m}(\tau) = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}(\tau)$ which through time differentiation becomes $\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \dot{\vec{\sigma}}(\tau)$. The derivative of the firing sequence $\boldsymbol{f}(\tau) = \dot{\vec{\sigma}}(\tau)$ 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 [19]. For a broad class of Petri nets it is shown that infinite server semantics offers better approximation than finite server semantics [15]. This paper deals with infinite server semantics for which the flow of a transition t_j at time τ is the product of the firing rate, λ_j, and the enabling degree of the transition at $\boldsymbol{m}(\tau)$

$$f(t_j, \tau) = \lambda_j \cdot enab(t_j, \boldsymbol{m}(\tau)) = \lambda_j \cdot \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m(p_i, \tau)}{Pre(p_i, t_j)} \right\} \tag{2.1}$$

For the sake of clarity, τ will be omitted in the rest of the report when there is no confusion: $f(t_j)$, $\boldsymbol{m}$ and $m(p_i)$ will be used instead of $f(t_j, \tau)$, $\boldsymbol{m}(\tau)$ and $m(p_i, \tau)$.

2.2 Implicit places and continuous marked graphs

A place p is called *implicit* when it is never the unique place restricting the firing of its output transitions. Hence, an implicit place can be removed without affecting the behavior of the rest of the system, i.e., the language of firing sequences of the original system is preserved [18].

Normally, implicit places are decided by the structure but also depend on their initial markings. The places that can be made implicit (with a proper initial marking) for any initial marking of the rest of the system are called *structurally implicit places*. For the subclass of structurally implicit places whose minimal initial marking can be deduced from the marking of other places is said to be *marking structurally implicit places*, which is formally characterized as the following:

Definition 2.2.1. *[20] Let $\mathcal{N} = \langle P \cup p, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$. The place p is* marking structurally implicit place*, iff there exists $\boldsymbol{y} \geq 0$, such that $\boldsymbol{C}(p, T) = \boldsymbol{y} \cdot \boldsymbol{C}(P, T)$.*

For strongly connected marked graphs, a marking structurally implicit place p verifies:

$$C(p, \cdot) = \sum_{p_j \in \pi} C(p_j, \cdot) \text{ for } \forall \pi \in \mathcal{P}(t_e, t_s) \tag{2.2}$$

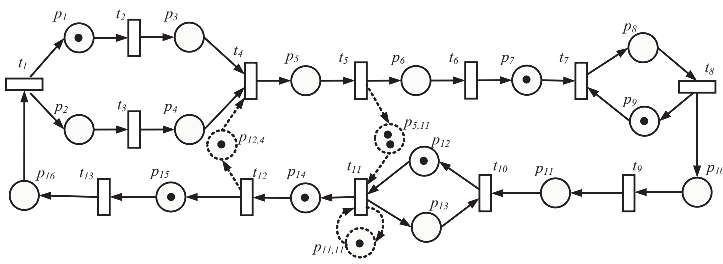


Figure 2.1: Marked Graph and Marking Structurally Implicit Places, with initial marking $\boldsymbol{m}_0(p_1) = \boldsymbol{m}_0(p_7) = \boldsymbol{m}_0(p_9) = \boldsymbol{m}_0(p_{12}) = \boldsymbol{m}_0(p_{14}) = \boldsymbol{m}_0(p_{15}) = 1$.

where t_e, t_s is input and output transition of p respectively, $\mathcal{P}(t_e, t_s)$ is the set of simple paths (i.e., the paths without repeated node) from t_e to t_s [6].

It is proved in [6] that, given a marking structurally implicit place $p_{e,s}$, the minimal initial marking to make $p_{e,s}$ implicit is:

$$m_0(p_{e,s}) = m_0^{min}(p_{e,s}) = \min \left\{ \sum_{p_j \in \pi} m_0(p_j) | \pi \in \mathcal{P}(t_e, t_s) \right\} \tag{2.3}$$

Example 2.2.1. *Fig. 2.1 shows a marked graph. It is easy to observe that from t_{12} to t_4 there exist two simple paths, $\pi_1 = \{t_{12}p_{15}t_{13}p_{16}t_1p_1t_2p_3t_4\}$, $\pi_2 = \{t_{12}p_{15}t_{13}p_{16}t_1p_2t_3p_4t_4\}$. Therefore, $\mathcal{P}(t_{12}, t_4) = \{\pi_1, \pi_2\}$.*

With respect to $\mathcal{P}(t_{12}, t_4)$, the added place $p_{12,4}$ is marking structurally implicit with input transition t_{12} and output transition t_4. Similarly, if considering the path from t_5 to t_{11}, $\pi_3 = \{t_5p_6t_6p_7t_7p_8t_8p_{10}t_9p_{11}t_{10}p_{12}t_{11}\}$, $p_{5,11}$ is the corresponding marking structurally implicit place. There is a loop path from t_{11}, $\pi_4 = \{t_{11}, p_{13}, t_{10}, p_{12}, t_{11}\}$, therefore $p_{11,11}$ is constructed which forms a self loop.

In order to compute minimal initial marking to make $p_{12,4}$ implicit, the sum of markings in each path from t_{12} to t_4 is considered. Because the sum of markings of places in π_1 is 2, while for π_2 it is 1, according to (2.3), the minimal is chosen, so one token should be put into $p_{12,4}$. Similarly, two tokens in $p_{5,11}$, and one token in $p_{11,11}$

When the net system is considered as continuous, the minimal intial marking makings of marking structurally implicit places can also be calculated using (2.3). The similar proof can be constructed as in [6], just noticing the fact that in continuous marked graphs a transition is not fireable iff one of its input place is empty, which is the same for the discrete ones.

Chapter 3

Minimum-time Control for Structurally Persistent *ContPN*

3.1 Problem Statement

We now consider net systems subject to external control actions, and assume that the only admissible control law consists in *slowing down* the firing speed of transitions [19]. Under this assumption, the controlled flow of a *ContPN* system is denoted as: $\boldsymbol{w}(\tau) = \boldsymbol{f}(\tau) - \boldsymbol{u}(\tau)$, with $0 \leq \boldsymbol{u}(\tau) \leq \boldsymbol{f}(\tau)$. The overall behavior of the system is ruled by: $\dot{\boldsymbol{m}} = \boldsymbol{C} \cdot (\boldsymbol{f}(\tau) - \boldsymbol{u}(\tau))$. In this work, we assume that every transition is *controllable* (t_j is uncontrollable if the only control that can be applied is $u(t_j) = 0$).

The problem we deal with in this work is: how to design a control action $\boldsymbol{u}$ that drives the system from the initial marking $\boldsymbol{m}_0$ to the desired final marking $\boldsymbol{m}_f$ in minimum-time?

Example 3.1.1. *Fig. 3.1 shows a structurally persistent and unbounded ContPN. Assume* $\boldsymbol{m}_0 = [1\ 0\ 0\ 0\ 0]^T$, $\boldsymbol{\lambda} = [1\ 1\ 1\ 1]^T$, *and the desired final marking* $\boldsymbol{m}_f = [0.3\ 0.4\ 0.3\ 0.4\ 0.4]^T$.

Considering the model as untimed, the following firing sequence ensures the reachability of the final marking: $\sigma = t_1(0.8)t_2(0.5)t_3(0.5)t_4(0.1)$. *Looking at the system as timed and considering* σ, *one can think to reach the final marking in the following way:*

(i) fire first as fast as possible t_1 *and stop the other transitions; since* $\int_0^{1.61} f(t_1)d\tau = 0.8$, *this firing takes* 1.61 *time units;*

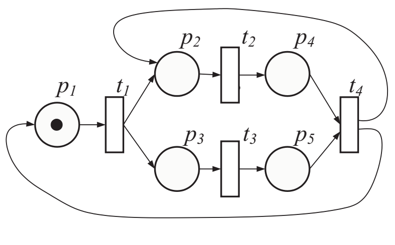


Figure 3.1: Structurally Persistent Petri Net System

(ii) *open t_2 until the integral of its flow is equal to 0.5 and stop the other transitions; this firing takes 0.98 time unit because $\int_0^{0.98} f(t_2)d\tau = 0.5$;*

(iii) *stop all transitions and fire only t_3 until its flow integral is 0.5; this will take 0.98 time unit because $\int_0^{0.98} f(t_3)d\tau = 0.5$;*

(iv) *finally, open only t_4 for 0.22 time unit because $\int_0^{0.22} f(t_4)d\tau = 0.1$.*

The previous strategy on the time system corresponds to the following control actions $\boldsymbol{u}(\tau)$:

(i) $\boldsymbol{u}(\tau) = [0\ f(t_2)\ f(t_3)\ f(t_4)]^T$ *for* $0 \leq \tau \leq 1.61$*;*

(ii) $\boldsymbol{u}(\tau) = [f(t_1)\ 0\ f(t_3)\ f(t_4)]^T$ *for* $1.61 < \tau \leq 2.59$*;*

(iii) $\boldsymbol{u}(\tau) = [f(t_1)\ f(t_2)\ 0\ f(t_4)]^T$ *for* $2.59 < \tau \leq 3.57$*;*

(iv) $\boldsymbol{u}(\tau) = [f(t_1)\ f(t_2)\ f(t_3)\ 0]^T$ *for* $3.57 < \tau \leq 3.79$*;*

(v) *if* $\tau > 3.79$, $\boldsymbol{u}(\tau) = \boldsymbol{f}(\tau)$*, i.e., all transitions are stopped.*

With this control actions, the system can reach the final marking in 3.79 time units, but as it will be shown, this is not a minimum-time controller because actions are unnecessarily sequentialized.

3.2 Minimum-time Controller

In this section, an *ON-OFF* controller is proposed for structurally persistent *ContPN* systems and it will be shown that it is a minimum-time controller. We will first present some assumptions, then the controller is designed for both discrete-time and continuous-time *ContPN*.

3.2.1 Minimal Firing Count Vector

In general, a marking $\boldsymbol{m}$ can be reached from $\boldsymbol{m}_0$ by using different firing sequences. For example, if the net is consistent and $\boldsymbol{m}$ is reached with σ, it is also reached when firing a T-*semiflow* $\alpha \geq 0$ times before (or interleaved with) σ. Here we introduce the notion of *minimal firing count vector*, and prove that it is unique under some assumptions for persistent nets.

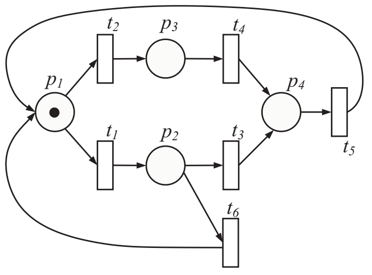


Figure 3.2: A non Structurally Persistent Petri Net System

Definition 3.2.1. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a ContPN system and $\boldsymbol{m}_f$ be a reachable marking through a sequence σ, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}$. The firing count vector $\vec{\sigma}$ is said to be* minimal *if for any T*-semiflow *$\boldsymbol{x}$, $||\boldsymbol{x}|| \not\subseteq ||\vec{\sigma}||$, where $|| \cdot ||$ stands for the support of a vector, i.e., the index of the elements different than zero.*

Example 3.2.1. *Let us consider the net system in Fig. 3.2 that is not structurally persistent because $p_1^\bullet = \{t_1, t_2\}$, and $p_2^\bullet = \{t_3, t_6\}$. Assume $\boldsymbol{m}_0 = [1\ 0\ 0\ 0]^T$ and $\boldsymbol{m}_f = [0\ 0\ 0\ 1]^T$. Firing the sequence $\sigma_1 = t_1(1)t_3(1)$ ($\vec{\sigma}_1 = [1\ 0\ 1\ 0\ 0\ 0]^T$) from $\boldsymbol{m}_0$ the obtained marking is $\boldsymbol{m}_f$. The same marking is obtained by firing $\sigma_2 = t_1(1)t_6(1)t_1(1)t_3(1)$ ($\vec{\sigma}_2 = \vec{\sigma}_1 + [1\ 0\ 0\ 0\ 0\ 1]^T$) since $[1\ 0\ 0\ 0\ 0\ 1]^T$ is a T*-semiflow*. Therefore, $\vec{\sigma}_1$ is a minimal firing count vector, while $\vec{\sigma}_2$ is not. Normally, the minimal firing count vector is not unique. For this net, $\vec{\sigma}_3 = [0\ 1\ 0\ 1\ 0\ 0]^T$ ($\sigma_3 = t_2(1)t_4(1)$) is another minimal firing count vector leading to $\boldsymbol{m}_f$.*

Proposition 3.2.1. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a structurally persistent PN system and $\boldsymbol{m}_f$ be a reachable marking. If one of the following assumption is satisfied, there exits a unique minimal firing count vector $\vec{\sigma}$.*

(A1) The matrix $\boldsymbol{C}$ has full rank;

(A2) The ContPN is strongly connected and consistent.

Proof. Suppose there exist two minimal firing count vectors $\vec{\sigma}_1$ and $\vec{\sigma}_2$, then (1) $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}_1$, (2) $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \vec{\sigma}_2$. Subtracting (2) from (1), we obtain:

$$\boldsymbol{C} \cdot (\vec{\sigma}_1 - \vec{\sigma}_2) = \boldsymbol{C} \cdot \vec{\sigma}_{12} = 0$$

If (A1) is satisfied, we must have $\vec{\sigma}_{12} = 0$, so $\vec{\sigma}_1 = \vec{\sigma}_2 (\neq 0$, if $m_f \neq m_0)$.

If (A2) is satisfied, there is only one minimal T-*semiflow* [22], denoted by $\boldsymbol{x} > 0$. $\vec{\sigma}_{12}$ may have negative elements, but we can always find an $\alpha \geq 0$, such that $\vec{\sigma}_{12} + \alpha \cdot \boldsymbol{x} \geq 0$. Since $\boldsymbol{C} \cdot (\vec{\sigma}_{12} + \alpha \cdot \boldsymbol{x}) = 0$ and $\vec{\sigma}_{12} + \alpha \cdot \boldsymbol{x} \geq 0$, it is a T-*semiflow.* Therefore, there exists $\beta > 0$ such that $\vec{\sigma}_{12} + \alpha \cdot \boldsymbol{x} = \beta \cdot \boldsymbol{x}$, implying $\vec{\sigma}_{12} = (\beta - \alpha) \cdot \boldsymbol{x}$. If $\beta - \alpha = 0$ then $\vec{\sigma}_1 = \vec{\sigma}_2$

which is impossible by assumption. If $\beta - \alpha > 0$ then $\vec{\sigma}_1 = \vec{\sigma}_2 + (\beta - \alpha) \cdot \boldsymbol{x} > (\beta - \alpha) \cdot \boldsymbol{x}$. Therefore, $\vec{\sigma}_1$ is not a minimal firing count vector. Similarly, if $\beta - \alpha < 0$ then $\vec{\sigma}_2$ is not a minimal firing count vector. □

Hence, for a structurally persistent system under assumption (A1) or (A2), any controller driving the system to $\boldsymbol{m}_f$ must follow *the* minimal firing count vector plus eventually a T-*semiflow*. If we are interested in the minimum-time controller then it should follow *only* the minimal firing count vector since the firing of a T-*semiflow* will be fired independently, before the minimal firing sequence and this firing takes time. In the following we will show that among all possible controllers having the integral of firing flow equal to the minimal firing count vector, the one corresponding to the *ON-OFF* strategy provides the minimum-time controller.

3.2.2 Minimum-time controller: Discrete-time Case

Sampling the continuous-time *ContPN* system with a *sampling period* Θ, we obtain the discrete-time *ContPN* [13] given by:

$$\begin{aligned} &\boldsymbol{m}(k+1) = \boldsymbol{m}(k) + \boldsymbol{C} \cdot \boldsymbol{w}(k) \cdot \Theta \\ &0 \leq \boldsymbol{w}(k) \leq \boldsymbol{f}(k) \end{aligned} \tag{3.1}$$

Here $\boldsymbol{m}(k)$ and $\boldsymbol{w}(k)$ are the marking and controlled flow at sampling instant k, i.e., at $\tau = k \cdot \Theta$. Let $u(t_j, k)$, $f(t_j, k)$ and $w(t_j, k)$ denote the control action, flow and controlled flow of transition t_j. The firing count of t_j in k^{th} sampling period is denoted by $s_k(t_j) = w(t_j, k) \cdot \Theta$.

Property 2.1.1 shows that for structurally persistent systems if two transitions t_1 and t_2 are enabled at the same time, the order of firing is not important (i.e., both sequences $t_1 t_2$ and $t_2 t_1$ are fireable).

Example 3.2.2. *Let us consider again the net system in Fig. 3.1 but now as discrete-time with* $\Theta = 0.2$. *Assume* $\boldsymbol{m}_0 = \boldsymbol{m}(0) = [0\ 1\ 1\ 1\ 1]^T$, $\boldsymbol{\lambda} = [1\ 1\ 1\ 1]^T$ *and* $\boldsymbol{m}_f = [0.2\ 1.1\ 0.9\ 0.9\ 0.9]^T$. *The minimal firing count vector in this case is* $\vec{\sigma} = [0\ 0.1\ 0.1\ 0.2]^T$. *The following controlled flow ensures the reaching of* $\boldsymbol{m}_f$ *in two sampling periods:*

- *At* $k = 0$*:* $w(t_1, 0) = w(t_4, 0) = 0$ *and* $w(t_2, 0) = w(t_3, 0) = 0.5$. *Then* t_2, t_3 *are fired in an amount* $0.5 \cdot \Theta = 0.1$ *and* t_1 *and* t_4 *are stopped. The system reaches* $\boldsymbol{m}(1) = [0\ 0.9\ 0.9\ 1.1\ 1.1]^T$.

- *At* $k = 1$*:* $w(t_1, 1) = w(t_2, 1) = w(t_3, 1) = 0$ *and* $w(t_4, 1) = 1$. *Then* t_1, t_2 *and* t_3 *are stopped while* t_4 *is fired in an amount* $1 \cdot \Theta = 0.2$. *After this sampling period,* $\boldsymbol{m}_f$ *is reached.*

Under this control law, t_2 and t_3 are fired before t_4. Since t_4 is 1-enabled at $\boldsymbol{m}_0$ it can be fired first and $\boldsymbol{m}_f$ is still reached. Therefore, another control law corresponding to the same minimal firing count vector is:

- *At $k = 0$: $w(t_1, 0) = w(t_2, 0) = w(t_3, 0) = 0$ and $w(t_4, 0) = 1$. Hence, t_1, t_2 and t_3 are stopped, and t_4 is fired in an amount $1 \cdot \Theta = 0.2$. Now, $\boldsymbol{m}(1) = [0.2\ 1.2\ 1\ 0.8\ 0.8]^T$.*

- *At $k = 1$: $w(t_1, 1) = w(t_4, 1) = 0$ and $w(t_2, 1) = w(t_3, 1) = 0.5$. Hence, t_2 and t_3 are fired in an amount 0.1 while t_1 and t_4 are stopped.*

Definition 3.2.2. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \Theta, \boldsymbol{m}_0 \rangle$ be a discrete-time ContPN system and $\boldsymbol{m}_f$ be a reachable final marking with a firing count vector $\vec{\sigma}$. Then, transition t_j is said to be* sufficiently fired *in the k^{th} sampling period if one of the following conditions holds:*

- $s_k(t_j) \overset{def}{=} w(t_j, k) \cdot \Theta = f(t_j, k) \cdot \Theta$, *i.e.*, $u(t_j, k) = 0$,

- $0 < s_k(t_j) \leq f(t_j, k) \cdot \Theta$ *and* $s_k(t_j) + \sum_{i=0}^{k-1} s_i(t_j) = \vec{\sigma}(t_j)$.

In the first case t_j is fired in the maximal amount, while in the second case it is the last firing of t_j in the corresponding sequence.

For instance, let us examine the first control law in Ex. 3.2.2 for $k = 0$. Transition t_4 is not sufficiently fired, because $\vec{\sigma}(t_4) = 0.2$ (should fire in an amount of 0.2) but $s_0(t_4) = w(t_4, 0) \cdot \Theta = 0$ (it is not fired). On the other hand t_2 and t_3 are sufficiently fired at $k = 0$ because $s_0(t_2) = s_0(t_3) = 0.1 = \vec{\sigma}(t_2) = \vec{\sigma}(t_3)$.

In control theory, an *ON-OFF* controller is a controller that switches abruptly between two states. It is frequently used in minimum-time problems and actually optimal control in many cases [16, 9]. Here we design an *ON-OFF* controller for structurally persistent Petri nets and prove that it is the minimum-time controller for such systems.

Definition 3.2.3. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \Theta, \boldsymbol{m}_0 \rangle$ be a structurally persistent discrete-time ContPN system and $\boldsymbol{m}_f$ be a reachable final marking with the corresponding minimal firing count vector $\vec{\sigma}$. An* ON-OFF *controller is defined as:* $u(t_j, k) =$

$$\begin{cases} 0 & \text{if } \sum_{i=0}^{k-1} s_i(t_j) + f(t_j,k)\cdot\Theta \le \vec{\sigma}(t_j) \quad (a) \\ f(t_j,k) & \text{if } \sum_{i=0}^{k-1} s_i(t_j) = \vec{\sigma}(t_j) \quad (b) \\ f(t_j,k) - \frac{\vec{\sigma}(t_j)-\sum_{i=0}^{k-1} s_i(t_j)}{\Theta} & \\ & \text{if } \sum_{i=0}^{k-1} s_i(t_j) < \vec{\sigma}(t_j) \text{ and } \quad (c) \\ & \sum_{i=0}^{k-1} s_i(t_j) + f(t_j,k)\cdot\Theta > \vec{\sigma}(t_j) \end{cases} \tag{3.2}$$

Assuming at $k = 0$, $\sum_{i=0}^{k-1} s_i(t_j) = 0$.

(a) says that before reaching the required total firing count $\vec{\sigma}(t_j)$*, we simply let transition* t_j *to fire* free*, i.e.* $u(t_j,k) = 0$*; (b) means once* $\vec{\sigma}(t_j)$ *is reached, the transition is completely stopped, i.e.* $u(t_j,k) = f(t_j,k)$*; (c) represents the last firing of* t_j *but cannot fire at maximum speed to not overpass* $\vec{\sigma}(t_j)$*. After all the transition are stopped, the system will stay in the final marking.*

The basic idea of the *ON-OFF* controller is that in each sampling period k, every transition is sufficiently fired.

Proposition 3.2.2. *Let* $\langle \mathcal{N}, \boldsymbol{\lambda}, \Theta, \boldsymbol{m}_0\rangle$ *be a structurally persistent discrete-time ContPN system and* $\boldsymbol{m}_f$ *be a reachable final marking with the corresponding minimal firing count vector* $\vec{\sigma}$*. The* ON-OFF *controller is the minimum-time controller driving the system to* $\boldsymbol{m}_f$*.*

Proof. We will prove that whenever there exists a controller $\boldsymbol{G}$ driving the system to $\boldsymbol{m}_f$, it consumes at least the time of the *ON-OFF* controller. This will imply that the *ON-OFF* controller is the minimum-time controller.

Assume a non *ON-OFF* controller $\boldsymbol{G}$. Hence, there exists a transition t_j that is not sufficiently fired in a sampling period k. In other words, t_j has to be fired later in a sampling period l, $l > k$. Let us assume, without loss of generality, that t_j is not fired between the k^{th} and the l^{th} sampling period. It is always possible to “move” some amount of firing from the l^{th} sampling period to the k^{th} one until t_j becomes sufficiently fired in k. According to Property 2.1.1 this move does not affect the fireability of the other transitions. Iterating the procedure, all transitions can be made sufficiently fired in all sampling periods and the obtained controller is an *ON-OFF* one.

Obviously, the number of discrete-time periods necessary to reach the final marking after moving firings from a sampling period l to another one k with $k \le l$ is at least the same. Hence the number of sampling steps is not higher with the *ON-OFF* controller. □

3.2.3 Minimum-time controller: Continuous-time Case

Definition 3.2.4. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a structurally persistent continuous-time ContPN system, and $\boldsymbol{m}_f$ be a reachable final marking with the minimal firing count vector $\vec{\sigma}$. An* ON-OFF *controller is defined as:*

$$u(t_j) = \begin{cases} 0 & \text{if } \int_0^{\tau^-} w(t_j) \mathrm{d}\tau < \vec{\sigma}(t_j) \quad (a) \\ f(t_j) & \text{if } \int_0^{\tau^-} w(t_j) \mathrm{d}\tau = \vec{\sigma}(t_j) \quad (b) \end{cases} \tag{3.3}$$

(a) means that if $\vec{\sigma}(t_j)$ is not reached then t_j is completely ON, *i.e., $u(t_j) = 0$; else (b), t_j is completely* OFF, *i.e., $u(t_j) = f(t_j)$.*

Corollary 3.2.1. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a structurally persistent continuous-time ContPN system, and $\boldsymbol{m}_f$ be a reachable final marking with the corresponding minimal firing count vector $\vec{\sigma}$. The* ON-OFF *controller given by 3.3 is the minimum-time one driving the system to $\boldsymbol{m}_f$.*

Proof. If we take sampling period $\Theta \to 0$ in Def. 3.2.3, the *ON-OFF* controller in Def. 3.2.4 is obtained. According to Proposition 3.2.2, this is the minimum-time controller. □

Let us notice that once a place of a continuous-time *ContPN* is marked, it will take infinite time to be emptied (like the discharging of a capacitor in an electrical RC-circuit). Therefore, the *ON-OFF* controller of a structurally persistent net is the minimum-time controller if no place is emptied during the trajectory from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$. Otherwise, the final marking is reached at the limit, in infinite time. For example, to reach $\boldsymbol{m}_f = [0\ 0\ 1]^T$ in the net system in Fig. 3.3, p_1 has to be emptied while p_2 should be marked first and then emptied. Hence, $\boldsymbol{m}_f$ is reached at the limit. Nevertheless, if $\boldsymbol{m}_f = [0.5\ 0.5\ 0]^T$ (that is not strictly positive), can be reached in finite time since there exists a trajectory no emptying any place, i.e., firing t_1 in an amount 0.5.

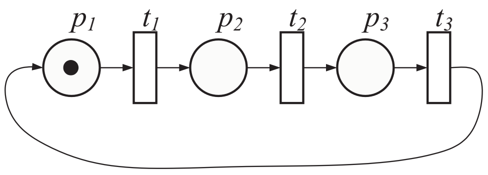


Figure 3.3: *ContPN* with $\boldsymbol{m}_0 = [1\ 0\ 0]$. Assume $\boldsymbol{m}_f = [0\ 0\ 1]$.

Example 3.2.3. *Let's consider the same problem in Ex. 3.1.1 (Fig. 3.1) but with an* ON-OFF *controller. The marking $\boldsymbol{m}_f$ is reached in* 1.65 *time units comparing with* 3.79 *time units in Ex. 3.1.1 where a different (pure sequentialized) controller with the same minimal firing count vector is applied. The marking trajectory is in Fig. 3.4.*

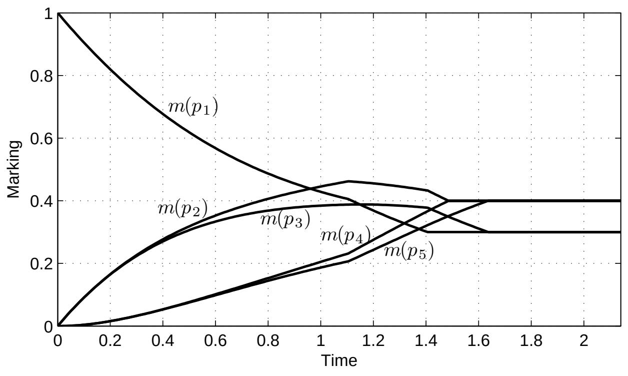


Figure 3.4: Marking trajectories

3.3 Case Study

Let's consider the net system in Fig. 3.5, which models a table factory system (taken from [22]).

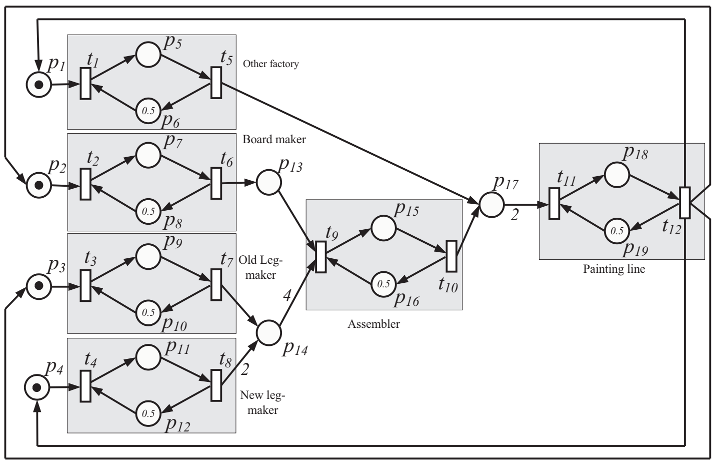


Figure 3.5: Persistent PN model of a table factory system. Assume the firing rate of every transition to be equal to 1.

The system consists of several parts, including board maker, leg maker, assembler, painting line and is modeled by a weighted structurally persistent $ContPN$. Suppose in the initial marking $m_0(p_1) = m_0(p_2) = m_0(p_3) = m_0(p_4) = 1$, $m_0(p_6) = m_0(p_8) = m_0(p_{10}) = m_0(p_{12}) = m_0(p_{16}) = m_0(p_{19}) = 0.5$, and the other places are empty. Assume $\boldsymbol{m}_f$ be $m_f(p_3) = m_f(p_{17}) = 0.1$, $m_f(p_4) = m_f(p_5) = 0.2$, $m_f(p_{13}) = 0.15$, and all the other places with marking equal to 0.25. The corresponding minimal firing count vector $\vec{\sigma} = [0.85\ 0.85\ 1.0\ 0.9\ 0.6\ 0.6\ 0.75\ 0.65\ 0.45\ 0.2\ 0.35\ 0.10]$.

Applying *ON-OFF* on the system under continuous time, Fig. 3.6 shows the stopping time instants of transitions. After t_9 is stopped at 4.28 time units, all the places are at the final state value.

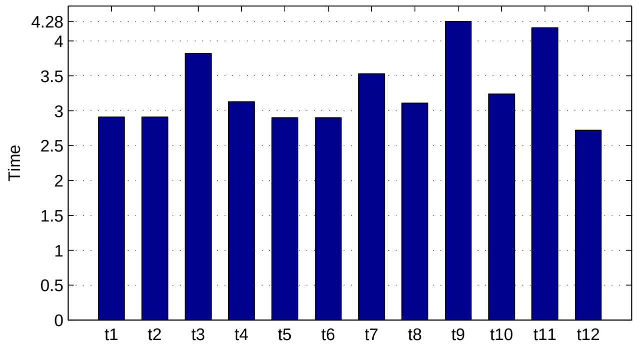


Figure 3.6: Stopping time instants

Fig. 3.7 shows the marking trajectory of place p_3, p_{13}, p_{14} and p_{17}. Taking p_{17} as a example, it reaches the final state at 4.19 time units. That makes sense, because the marking of p_{17} is dependent on transitions t_5, t_{10} and t_{11}, which are stopped at 2.9, 3.24 and 4.19 time units, respectively. When t_{11} is stopped, the system has reached the final state.

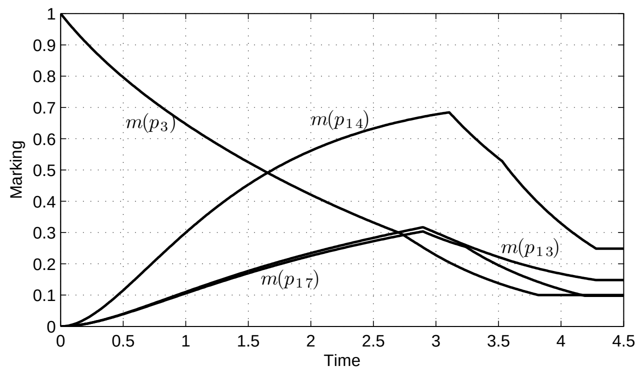


Figure 3.7: Marking trajectories

Chapter 4

Distributed Control of Large Scale Systems

In this chapter, we address how to apply the minimum-time *ON-OFF* contorlle to distributed control of large scale systems. The more detailed content of this chapter can be found in Appendix A([23]).

4.1 Problem Statement

The classical centralized control theory has been proved inefficient for large scale distributed systems, in which the communication delay, time synchronization problems become significant. Therefore distributed or decentralized control is extensively explored in recent decades. In a distributed controlled system, normally a complex dynamic system, the controllers are not centralized in one location, but are distributed in the subsystems, while typically, each controller can only access local resources and limited information from its neighbor subsystems.

Under the framework of *ContPN*, the large scale system is decomposed into subsystems that are modeled with *ContPNs* and controlled by the local controllers. Each local controller can obtain informations from its neighbor subsystems through interface places and transitions. The problem we deal with in this chapter is: how to design the control action for each local controller which works independently, and drives the system from initial marking m_0 to final marking m_f (in minimum-time). Large net systems are first decomposed according to sets of places, after that, adequate control actions for using *ON-OFF* controller are computed locally.

4.2 Structural Decomposition of Marked Graphs

In this section we adapt the decomposition methods developed in [6]. The idea is the following: given a strongly connected marked graph $\mathcal{N}$, it is first split into two subnets $\mathcal{N}_1$ and $\mathcal{N}_2$ according to a set of places $B \subseteq P$, after that the *complemented subnets* ($\mathcal{CN}$) are derived through adding interface transitions and marking structurally implicit places.

Definition 4.2.1. *Let* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *be a strongly connected marked graph,* $B \subseteq P$ *is said to be a* cut *iff there exists subnet* $\mathcal{N}_i = \langle P_i, T_i, \boldsymbol{Pre}_i, \boldsymbol{Post}_i \rangle$, $i = 1, 2$, *such that:*

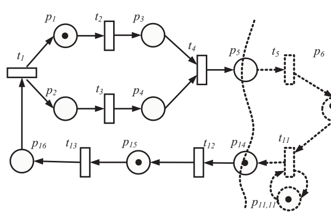
(i) $T_1 \cup T_2 = T$, $T_1 \cap T_2 = \emptyset$

(i) $P_1 \cup P_2 = P$, $P_1 \cap P_2 = B$

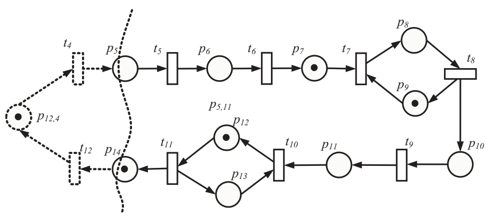
(ii) $P_1 =^\bullet T_1 \cup T_1^\bullet$, $P_2 =^\bullet T_2 \cup T_2^\bullet$

where $U =^\bullet B \cup B^\bullet$ is said to be interface, which is partitioned into U_1, U_2, such that $U_1 \cup U_2 = U$, $U_i = T_i \cap U$.

Example 4.2.1. $\mathcal{N}_1$, $\mathcal{N}_2$ *(the dotted part is not included) in Fig. 4.1 are the subnets obtained from the marked graph in Fig. 2.1, which is cut by* $B = \{p_5, p_{14}\}$, *with the interface* $U = \{t_4, t_5, t_{11}, t_{12}\}$ *and* $U_1 = \{t_4, t_{12}\}$, $U_2 = \{t_5, t_{11}\}$.



(a) $\mathcal{N}_1$(dotted part not included) and $\mathcal{CN}_1$(dotted part included)



(b) $\mathcal{N}_2$(dotted part not included) and $\mathcal{CN}_2$(dotted part included)

Figure 4.1: Cutting of marked graph

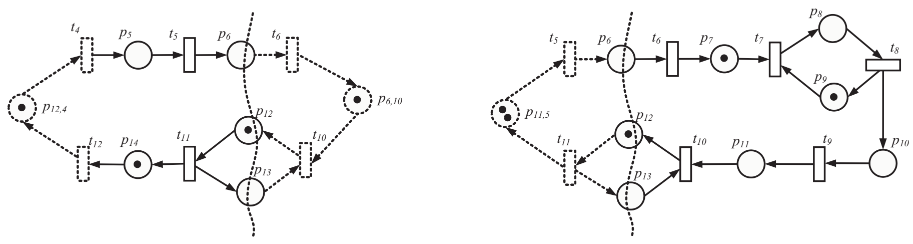
After cutting, the two subsystems $\mathcal{N}_1$, $\mathcal{N}_2$ are independent of the rest of system, because all the constraints from the rest of the system are removed, therefore different behaviors are introduced. The *complemented subnet* is obtained after adding marking structurally implicit places as approximations of other parts of the system.

Definition 4.2.2. *Let* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *be a strongly connected marked graph,* $\mathcal{N}_i = \langle P_i, T_i, \boldsymbol{Pre}_i, \boldsymbol{Post}_i \rangle$ *be the subnets associated with a cut* B. *The* complemented subnet $\mathcal{CN}_i$ *is obtained from* $\mathcal{N}_i$ *by adding transitions in* U_j *and the marking structurally implicit places with respect to the paths* $\mathcal{P}(t_e, t_s)$ *in* $\mathcal{N}_j$, $t_e, t_s \in U_j$, $i, j = 1, 2, i \neq j$. *The set of places being added to* $\mathcal{N}_i$ *is denoted by* IP_i .

In Fig. 4.1, the complemented subnet $\mathcal{CN}_1$ is obtained after adding $U_2 = \{t_5, t_{11}\}$ and $IP_1 = \{p_{5,11}, p_{11,11}\}$ to $\mathcal{N}_1$, while $\mathcal{CN}_2$ is obtained after adding $U_1 = \{t_4, t_{12}\}$ and $IP_2 = \{p_{12,4}\}$ to $\mathcal{N}_2$. Notice that cut B and interface U are shared in both subnets.

In order to calculate the initial marking of $p_{e,s}$ making it implicit, we have to find out the *minimal path* from t_e to t_s such that (2.3) is satisfied. There are some efficient algorithms developed in literatures which can be used to search the minimal path, for example the algorithm of Floyd-Warshall [1] with a computation complexity of $O(|T|^3)$, where $|T|$ is the number of transitions.

Sometimes for a complex system, only one cut is not sufficient, because the complemented subsets are still difficult to handle. Therefore, the above decomposition process needs to be executed in multiple hierarchical levels. Fig. 4.2 presents the complemented subnets obtained from cutting $\mathcal{CN}_2$ in Fig. 4.1(b) one more time, with $B = \{p_6, p_{12}, p_{13}\}$. After this two level cutting, the original system is decomposed into three: $\mathcal{CN}_1$, $\mathcal{CN}_{21}$ and $\mathcal{CN}_{22}$. It should be noticed that the order of cutting is not important, for example in this case, if it is first cut with $B = \{p_6, p_{12}, p_{13}\}$, then with $B = \{p_5, p_{14}\}$ the exactly same subnets are obtained.



(a) $\mathcal{CN}_{21}$ (b) $\mathcal{CN}_{22}$

Figure 4.2: Complemented subnets: second cut from $\mathcal{CN}_2$

4.3 Application of the minimum-time Controller in Distributed Control

Our goal is to decompose a large scale system into smaller subsystems, and drive the system to the final state with the local controllers. In this section we will show that if the system is decomposed with the methods we present in section 4.2, and applying *ON-OFF* to each subsystem, the final state can be reached in minimum time.

Because marked graphs is a subclass of structurally persistent nets, this *ON-OFF* strategy can be applied. The problem is in a large distributed system, each local controller can only access limited resources from its neighbors, so the global control law (minimal firing count vector) can not be obtained directly. In the following parts, it is presented how to calculate it in a distributed way.

The idea is the following: minimal firing count vector is first calculated in each complemented subnet, since these firing count vectors may not be fireable when considering the global state of the system, then algorithms are proposed for constructing the global minimal firing count vectors without knowing the structures of subsystems.

In the sequel, we denote $\vec{\boldsymbol{\sigma}}$ the minimal firing count vector driving $\mathcal{N}$ from $\boldsymbol{m}_0$ to $\boldsymbol{m}_f$ and $\vec{\boldsymbol{\sigma}}_i$ the firing count vector computed from $\mathcal{CN}_i$ for reaching $\boldsymbol{m}_f^i$ from $\boldsymbol{m}_0^i$, where $\boldsymbol{m}_0^i$ and $\boldsymbol{m}_f^i$ are initial and finial markings in $\mathcal{CN}_i$, projected from $\mathcal{CN}$.

Let us consider the case when the system is decomposed hierarchically. For a complemented subnet $\mathcal{CN}_i$, if $\vec{\boldsymbol{\sigma}}(t_j) = \vec{\boldsymbol{\sigma}}_i(t_j), t_j \in T_i$, then $\mathcal{CN}_i$ is said to be *critical.* Firing count vectors $\vec{\boldsymbol{\sigma}}_1$ and $\vec{\boldsymbol{\sigma}}_2$ are said to be *compatible* if $\vec{\boldsymbol{\sigma}}_1(t_j) = \vec{\boldsymbol{\sigma}}_2(t_j)$, $t_j \in U$. Define operator $\oplus$ the *merge* of $\vec{\boldsymbol{\sigma}}_1$, $\vec{\boldsymbol{\sigma}}_2$ if they are compatible, such that, $\vec{\boldsymbol{\sigma}}_{12} = \vec{\boldsymbol{\sigma}}_1 \oplus \vec{\boldsymbol{\sigma}}_2$, $\forall t \in T_i$, $\vec{\boldsymbol{\sigma}}_{12}(t) = \vec{\boldsymbol{\sigma}}_i(t)$, $i = 1, 2$.

Two complemented subnets are neighbors if they share a cut. Because every time we split one net into two parts, each complemented subnets has at least one neighbor. We will prove it is possible to make pairs of minimal firing vectors of neighbors to be compatible and obtain $\vec{\boldsymbol{\sigma}}$ after merging all of them.

Proposition 4.3.1. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a live marked graph, and decomposed into n subnets. Assuming $\mathcal{CN}_q$, $1 \leq q \leq n$ is critical complemented subnet, there exist $x_i, i = 1, 2, ..., n$ such that:*

$$\vec{\boldsymbol{\sigma}} = \bigoplus_{i=1}^{n} (\vec{\boldsymbol{\sigma}}_i + x_i \cdot \mathbf{1}) \tag{4.1}$$

and if $i = q, x_i = 0$, else $x_i \geq 0$.

Proof. See [23] □

It is clear $\vec{\boldsymbol{\sigma}}_i + x_i \cdot \mathbf{1}$ can be fired in $\mathcal{CN}_i$ and the corresponding final marking is reached. If applying the *ON-OFF* controller to the transitions in each subnet $\mathcal{N}_i$ with control action $\vec{\boldsymbol{\sigma}}_i + x_i \cdot \mathbf{1}$, the final state of system is approached in minimum-time. Now what needs to be done is designing an effective algorithm for searching a critical subnet, and calculating corresponding x_i.

Here we propose an algorithm for searching $\mathcal{CN}_q$ based on the graph $G = \langle V, W \rangle$, depicting the relations among complemented subnets, in which each node v represents a subnet, there are arcs between two nodes if the corresponding subnets are neighbors. The weight of the arc from v_i to v_j is $w(v_i, v_j) = \vec{\boldsymbol{\sigma}}_i(t) - \vec{\boldsymbol{\sigma}}_j(t), t \in U$, negative weight is also allowed here. Denote $W(v_i, v_j)$ the sum of the weights on the simple path from v_i to v_j.

Algorithm 1 Search a critical subnet

```
Input: $G = \langle V, W \rangle$
Output: A node $v_q \in V$
 1: Label all the nodes in $V$ as new;
 2: while more than one node in $V$ is labeled as new do
 3:    Choose a node $v_i$ from $V$ which is labeled as new;
 4:    for j = 1 to n do
 5:       if $W(j, i)$ has not been calculated then
 6:          calculate $W(i, j)$;
 7:          if $W(i, j) > 0$ then
 8:             label $v_j$ as old;
 9:          else if $W(i, j) < 0$ then
10:             label $v_i$ as old;
11:             break;
12:          end if
13:       end if
14:    end for
15:    if $j = n$ and $v_i$ is labeled as new then
16:       return $v_i$;
17:    end if
18: end while
19: return The last node in $V$ that is labeled as new
```

As for the calculation of x_i in (4.1), it is actually equal to the sum of weights of arcs in the path from node v_i to node v_q [23]. Then the overall procedure for the distributed control of a large scale system can be described as:

1: Structurally decomposition;
2: Construct the graph $G = \langle V, W \rangle$;
3: Search a critical subnet $\mathcal{CN}_q$ using Algorithm 1;
4: Compute $\boldsymbol{x}(i)$: the relative difference of $\mathcal{CN}_q$ to $\mathcal{CN}_i$;
5: Apply *ON-OFF* controller to $\mathcal{N}_i$ with control action $\vec{\boldsymbol{\sigma}}_i + \boldsymbol{x}(i) \cdot \mathbf{1}$.

Chapter 5

Conclusions

In this report, an *ON-OFF* controller is designed for structurally persistent *ContPN* systems which can drive the system from an initial marking to an final one in minimum-time. The idea behind is extremely simple and efficient: the final marking can be reached with different firing count vectors, but we only focus on the minimal one. In the framework of discrete-time systems, we design the *ON-OFF* controller, such that all transitions are fired as fast as possible, and suddenly stopped when the total firing counts are reached. Special attention should be paid to the last sampling period before stopping in order to prevent the total firing count to be exceed. It is proved that with this controller the final marking is reached in minimum-time. By considering the limit (going to zero) of the sampling period, the results are extended to continuous time systems.

Distributed control could be a solutions for control of systems that are too complex to handle with centralized controller or the deployments of systems are physically distributed. We focus on distributed control of large scale systems that are modeled with timed continuous marked graphs, aiming to drive the system from initial marking to desired final marking. The model is first decomposed into subnets with sets of places, then making structurally implicit places are introduced to obtain complemented subnets and control laws can be computed in a distributed way. After that, *ON-OFF* control is applied in each subnet, and final marking is reached in minimum time. One of the main future work will be applying this control method to more general nets structures, for example structurally persistent nets, where the decomposition method and approximation strategy should be reconsidered.

Appendix A

Distributed Control of Large Scale Systems Modeled with *ContPN*

Bibliography

[1] A. V Aho, J. D Ullman, and J. E Hopcroft. *Data Structures and Algorithms*. Addison Wesley, 1983.

[2] A. Amrah, N. Zerhouni, and A. El Moudni. On the control of manufacturing lines modelled by controlled continuous Petri nets. *Journal of Systems Science*, 29(2):127–137, 1998.

[3] H. Apaydin-Ozkan, J. Júlvez, C. Mahulea, and M. Silva. An Efficient Heuristics for Minimum Time Control of Continuous Petri nets. In *ADHS'09: 3^{rd} IFAC Conference on Analysis and Design of Hybrid Systems*, pages 44–49, Zaragoza, Spain, September 2009.

[4] H. Apaydin-Ozkan, J. Julvez, C. Mahulea, and M. Silva. A Control Method for Timed Distributed Continuous Petri nets. In *2010 American Control Conference*, Baltimore, USA, June 2010. to appear.

[5] F. Balduzzi, A. Giua, and C. Seatzu. Hybrid Control of Production Systems with Local Optimization. In *Proc. of the 7th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, pages 1531–1540, Barcelona, Spain, October 1999.

[6] J. Campos, J.M. Colom, H. Jungnitz, and M. Silva. *Approximate Throughput Computation of Stochastic Marked Graphs*. Springer, 1995.

[7] R. David and H.Alla. *Autonomous and timed continuous Petri nets*. Springer, Berlin, 2010.

[8] X. Guan and L. E. Holloway. Control of Distributed Discrete Event Systems Modeled as Petri Nets. In *Proceedings of the American Control Conference*, pages 2342–2347, Albuquerque, New Mexico, June 1997.

[9] H. Jang-Ho Robert Kim; Maurer. Sensitivity analysis of optimal control problems with bang-bang controls . In *42nd IEEE Conference on Decision and Control*, volume 4, pages 3281 – 3286, December 2003.

[10] J. Júlvez, L. Recalde, and M. Silva. On reachability in autonomous continuous Petri net systems. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Applications and Theory of Petri Nets 2003*, volume 2679, pages 221–240, Eindhoven, The Netherlands, June 2003. Springer Berlin / Heidelberg.

[11] P. Krishnan. Distributed timed automata. In *WDS'99, Workshop on Distributed Systems (A satellite workshop to FCT'99)*, volume 28 of *Electronic Notes in Theoretical Computer Science*, pages 5 – 21, 2000.

[12] L. H. Landweber and E L Robertson. Properties of conflict-free and persistent Petri nets. *J. ACM*, 25(3):352–364, 1978.

[13] 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.

[14] 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.

[15] 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.

[16] Dale B. McDonaldi. Structurally Time Optimal Control for Linear Systems with Bounded Control and Bias Disturbance. In *American Control Conference*, pages 1964 – 1969, New York, July 2007.

[17] R. P. Moreno, D. Tardioli, and J.l.V. Salcedo. Distributed Implementation of Discrete Event Control Systems based on Petri nets. In *ISIE08: IEEE International Symposium on Industrial Electronics*, pages 1738–1745, June 2008.

[18] M. Silva. *Las Redes de Petri en La Automática y la Informática.* Editorial AC, Madrid, 1985.

[19] 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.

[20] M. Silva, E. Teruel, and J.M. Colom. Linear algebraic and linear programming techniques for the analysis of P/T net systems. *LCNS*, 1:309–373, 1998.

[21] L. M. Sonneborn and F. S. Van Vleck. The Bang-Bang Principle for Linear Control Systems. In *J. Soc. Indus. and Appl. Math. Ser. A*, volume 2, pages 151–159, January 1964.

[22] E. Teruel, J. M. Colom, and M. Silva. Choice-free Petri nets: A model for deterministic concurrent systems with bulk services and arrivals. *IEEE Trans. on Systems, Man, and Cybernetics*, 27(1):73–83, 1997.

[23] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Distributed Control of Large Scale Systems Modeled with Continuous Petri Nets. 2010. report.

[24] J. Xu, L. Recalde, and M. Silva. Tracking control of join-free timed continuous Petri net systems under infinite server semantics. *Discrete Event Dynamic Systems*, 18(2):263–288, 2008.

[25] G. Yasuda. Design and implementation of Petri net based distributed control architecture for robotic manufacturing systems. In *MICAI'07: Proceedings of the artificial intelligence 6th Mexican international conference on Advances in artificial intelligence*, pages 1151–1161, Berlin, Heidelberg, 2007. Springer-Verlag.