TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2013

57

Liewei Wang

On Minimum-time Control of Continuous Petri nets: Centralized and Decentralized Perspectives

Departamento
Informática e Ingeniería de Sistemas

Director/es
Silva Suárez, Manuel
Mahulea, Cristian Florentín

ISSN 2254-7606

Prensas de la Universidad Universidad Zaragoza

Universidad Zaragoza
1542

Tesis Doctoral

ON MINIMUM-TIME CONTROL OF CONTINUOUS PETRI NETS: CENTRALIZED AND DECENTRALIZED PERSPECTIVES

Autor

Liewei Wang

Director/es

Silva Suárez, Manuel
Mahuelea Cristian, Florentín

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2013

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

On Minimum-time Control of Continuous Petri nets: Centralized and Decentralized Perspectives

Liewei WANG

Ph.D. Thesis

Advisor: Manuel Silva Suárez and Cristian Mahulea

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

May 2013

Resumen

Muchos sistemas artificales, como los sistemas de manufactura, de logística, de telecomunicaciones o de tráfico, pueden ser vistos "de manera natural" como Sistemas Dinámicos de Eventos Discretos (DEDS). Desafortunadamente, cuando tienen grandes poblaciones, estos sistemas pueden sufrir del clásico problema de la *explosión de estados*. Con la intención de evitar este problema, se pueden aplicar técnicas de fluidificación, obteniendo una relajación fluida del modelo original discreto. Las redes de Petri continuas (CPNs) son una aproximación fluida de las redes de Petri discretas, un conocido formalismo para los DEDS. Una ventaja clave del empleo de las CPNs es que, a menudo, llevan a una substancial reduccin del coste computacional.

Esta tesis se centra en el control de Redes de Petri continuas temporizadas (TCPNs), donde las transiciones tienen una interpretación temporal asociada. Se asume que los sistemas siguen una *semántica de servidores infinitos* (velocidad variable) y que las acciones de control aplicables son la *disminución de la velocidad* del disparo de las transiciones. Se consideran dos interesantes problemas de control en esta tesis: 1)*control del marcado objetivo*, donde el objetivo es conducir el sistema (tan rápido como sea posible) desde un estado inicial a un estado final deseado, y es similar al problema de control *set-point* para cualquier sistema de estado continuo; 2)*control del flujo óptimo*, donde el objetivo es conducir el sistema a un flujo óptimo sin conocimiento *a priori* del estado final. En particular, estamos interesados en alcanzar el flujo máximo tan rápido como sea posible, lo cual suele ser deseable en la mayoría de sistemas prcticos.

El problema de control del marcado objetivo se considera desde las perspectivas centralizada y descentralizada. Proponemos varios controladores centralizados en tiempo mínimo, y todos ellos están basados en una estrategia ON/OFF. Para algunas subclases, como las redes *Choice-Free* (CF), se garantiza la evolución en tiempo mínimo; mientras que para redes generales, los controladores propuestos son heurísticos. Respecto del problema de control descentralizado, proponemos en primer lugar un controlador descentralizado en tiempo mínimo para redes CF. Para redes generales, proponemos una aproximación distribuida del método *Model Predictive Control* (MPC); sin embargo en este método no se considera evolución en tiempo mínimo. El problema de control de flujo óptimo (en nuestro caso, flujo máximo) en tiempo mínimo se considera para redes CF. Proponemos un algoritmo heurístico en el que calculamos los "mejores" *firing count vectors* que llevan al sistema al flujo

máximo, y aplicamos una estrategia de disparo ON/OFF. También demostramos que, debido a que las redes CF son persistentes, podemos reducir el tiempo que tarda en alcanzar el flujo máximo con algunos disparos adicionales. Los métodos de control propuestos se han implementado e integrado en una herramienta para Redes de Petri híbridas basada en Matlab, llamada SimHPN.

Abstract

Many man-made systems, such as manufacturing, logistics, telecommunication or traffic systems, can be "naturally" viewed as Discrete Event Dynamic Systems (DEDS). Nevertheless, in the case of large populations they may suffer from the classical *state explosion* problem. In order to overcome this problem, *fluidization* can be applied, obtaining the fluid relaxation of the original discrete model. Continuous Petri nets (CPNs) are a fluid approximation of discrete Petri nets (PNs), a well known formalism for DEDS. One key benefit of using CPNs is that, most frequently, it leads to a substantial reduction in the computational costs.

In this thesis we focus on the control of timed continuous Petri nets (TCPNs), in which time interpretations are associated to transitions. We assume that net systems are under *infinite server semantics* (variable speed) and control actions are applied to *slow down* the firing of transitions. We consider two interesting control problems in this thesis: 1) *target marking control*, where the objective is to drive the system (as fast as possible) from an initial state to a desired final state, and it is similar to the *set-point* control problem in a general continuous-state system; 2) *optimal flow control*, in which the objective is to drive the system to an optimal flow, without *a priori* knowledge of a specific final state. In particular, we are interested in reaching as fast as possible the maximal flow, what is frequently desirable in practical systems.

The target marking control problem is considered in both centralized and decentralized settings. We propose several minimum-time centralized controllers and all of them are based on an ON/OFF strategy. For some subclasses like *Choice-Free* (CF) nets, minimum-time evolution is guaranteed; for general nets, the proposed controllers are heuristics. Regarding the decentralized control, we first propose a minimum-time decentralized controller for CF nets. Then, for general nets, we propose a distributed Model Predictive Control (MPC) approach; however, in this method, minimum-time evolution is not considered. The minimum-time optimal flow (in our case, the maximal flow) control problem is considered for CF nets. We propose a heuristic algorithm, in which we compute the "best" firing count vectors bringing the system to the maximal flow and an ON/OFF firing strategy is applied. We also show that because of the *persistency* of CF nets, we can further reduce the time spent to reach the maximal flow by means of some additional firings. The proposed control methods are implemented and integrated into a Matlab based toolbox for hybrid PN systems, called, SimHPN.

Contents

1 Introduction **1**

2 Continuous Petri nets: Basic Concepts and Notations **7**
2.1 (Discrete) Petri nets and the state explosion problem 8
2.2 Autonomous (untimed) continuous Petri nets 9
2.2.1 Basic concepts 10
2.2.2 Petri nets subclasses 10
2.2.3 Basic structural concepts 11
2.2.4 Reachability and lim-reachability 12
2.2.5 Boundedness 14
2.2.6 Liveness and deadlock-freeness 14
2.2.7 Implicit places and structurally implicit places 15
2.3 Timed continuous Petri nets 16
2.3.1 Conceptual framework and server semantics 16
2.3.2 Timed models versus untimed models 18
2.3.3 Performance bounds under infinite server semantics 19
2.3.4 Parametric optimization under infinite server semantics . . . 20
2.3.5 Approximation to the discrete systems 21
2.4 An example: a kanban-like manufacturing system 24

3 Control of Continuous Petri nets **31**
3.1 Introduction 32
3.1.1 Controlling the systems 32
3.1.2 Target marking control problem 32
3.1.3 Optimal flow control problem 34
3.2 Computing the initial and desired final states 34
3.2.1 About $\boldsymbol{m}_0$ 34
3.2.2 About the desired/final state 35
3.3 Controllability 36
3.4 Previous centralized control methods 38
3.4.1 Initial comparisons 42
3.5 Conclusions 46

4 Centralized Control: ON/OFF Based Methods **49**
4.1 Motivation: minimum-time state evolution 50
4.2 Minimum-time controller for Choice-Free nets 51
4.2.1 Minimal firing count vector 51
4.2.2 ON/OFF controller: discrete-time case 52
4.2.3 ON/OFF controller: continuous-time case 56
4.2.4 A case study 57
4.3 Drawbacks of the ON/OFF controller for general nets 59
4.4 Extended ON/OFF based methods 60
4.4.1 ON/OFF+ controller 60
4.4.2 Balanced ON/OFF controller (B-ON/OFF) 64
4.4.3 MPC-ON/OFF controller 68
4.4.4 Initial Comparisons 70
4.4.5 A case study 71
4.5 Computation of minimum-time control laws 74
4.6 Conclusions 75

5 Decentralized Control of CF nets: ON/OFF Based Methods **77**
5.1 Problem definitions 78
5.2 Related work 78
5.3 Decentralized control of CF nets 80
5.3.1 Cutting the system 80
5.3.2 Reduction rules 80
5.3.3 Complemented subsystems 88
5.3.4 The control law computation 90
5.3.5 A case study 93
5.4 Conclusions 94

6 Distributed MPC Control of General nets **97**
6.1 Introduction 98
6.2 A centralized MPC controller 99
6.3 Application to Distributed MPC control 103
6.3.1 Two subsystems 104
6.3.2 Multiple subsystems 109
6.3.3 A simple example 110
6.4 Conclusions 111

7 Minimum-time Flow Control of CF nets **113**
7.1 Motivations 114
7.2 Difficulties of Minimum-time Flow Control 115
7.3 A heuristic algorithm for CF nets 116
7.4 Examples 120
7.5 Conclusions 121

8 Simulations and Comparisons **123**
8.1 Implementation: SimHPN 124
8.2 Case study 1: centralized control of a manufacturing cell 126
8.3 Case study 2: centralized control of an assembly line 131
8.4 Case study 3: centralized control of a manufacturing system 134
8.5 Discussions of case study 1–3 135
8.6 Case study 4: distributed control of an AGV System 137
8.7 Conclusions 143

9 Conclusions and Future works **145**

A Simulation Results **161**

Chapter 1

Introduction

Petri nets (PNs) are a well known modelling paradigm initially introduced by C. A. Petri [76] as a fully *non-deterministic* (untimed) conceptual framework to logically model and analyze concurrency and synchronization in Discrete Event Dynamic Systems (DEDS). They have been widely applied in the industry for the analysis of manufacturing, traffic, or software systems, for example [61, 62, 71]. Some main features of PNs can be described as the following: (1) PNs are a graphical formalism that is able to depict, visually and straightforwardly, concurrency, conflict, synchronization, etc.; (2) PNs provide very compact representations of a system, enjoying a *bipartite* structure: places (as queues in *queueing networks*) are "containers" and transitions (as stations in queueing networks) are "activities"; (3) Different to other formalisms like *automata* or *Markov chains*, in which a symbolic unstructured global state is considered, in PNs the state is represented in a distributed and numerical way, in particular, as a vector of non-negative integers; (4) The locality of places (states) and transitions (the changes of states) facilitates both the *top-down* and *bottom-up* modelling methodologies. For instance, it is possible to *refine* a place or transition for a more detailed model; or *fuse* several places or transitions into one.

Nevertheless, similarly to other modelling formalisms, PNs also suffer from the *state explosion* problem, inherent to a large part of DEDS. In particular, the size of the state space may grow exponentially on the number of places and on the initial states. Therefore, the traditional state enumeration based methods may easily become intractable because of the high computational complexity. To overcome it, a classical relaxation technique called *fluidization* can be used. Continuous PNs (CPNs) are fluid approximations of classical discrete PNs, obtained by removing the integrality constraints. The firing count vectors and consequently the markings are no longer restricted to be in the naturals, but relaxed into non-negative real numbers. The idea of the fluidization of Petri nets was proposed first in 1987 in the field of manufacturing systems by David and Alla (see [25] for a comprehensive view), at the net level. Developed in paralell and very similarly, the fluidization at the level of the *state equation* was proposed at the same meeting (the 8th European Workshop on Application and Theory of PNs, Zaragoza) by Silva and Colom (see [90]), focus-

ing on the use of linear programming techniques to analyze the net systems. An important advantage of this relaxation is that more efficient algorithms are available for their analysis, at the price of losing some modelling or analysis capabilities, e.g. mutual exclusion, with respect to the discrete view (see [87] for a recent and broad survey). The discrete net system may also be partially fluidized. For instance, *First Order Hybrid PNs* were proposed and they can be used for optimization and control purposes [9, 26]. In [95, 39] stochastic PNs were extended to Fluid Stochastic PNs by introducing places with continuous tokens and arcs with fluid flow, in which the discrete and continuous portions may affect each other, so as to handle stochastic fluid flow systems. Moreover, fluidization is not a new technique, for example, it has also been extensively explored in *queueing networks* (see, for example, [72, 2, 16]).

Initially introduced as a fully non-deterministic model, the autonomous (untimed) PNs can be used to analyze logical properties of the system such as boundedness, liveness, etc. By introducing time to the model, we obtain timed PNs that are widely applied in performance evaluation and optimization. In the literature, time is associated mainly to transitions, which is also assumed in this thesis (other methods consist in associating time to the places or to the arcs, even to the tokens). Similarly, continuous PNs can also be autonomous or timed. Depending on how the *flow* of transitions is defined for timed continuous PNs (TCPNs), different *server semantics* appear. The *finite server semantics* (or constant speed) and *infinite server semantics* (or variable firing speed) [89, 25] are the most used ones. In this thesis, we focus on the *infinite server semantics*, since it has been proved that TCPNs under infinite server semantics approximate better the underlying discrete systems for a broad subclass of nets, under some general conditions [66]. The main topic of this thesis is the control of TCPNs, in both centralized and decentralized settings.

In Chapter 2 we recall the main definitions, concepts (such as reachability, boundedness, liveness, implicit places etc.) of continuous PNs, both for the autonomous (untimed) and timed models. We also introduce the main techniques for computing performance bounds and parametric optimizations using TCPNs. Usually, for populated systems continuous PNs can provide quite good fluid approximations to the underlying discrete systems (the reason can be partially understood by using the Functional *Central Limit* (Donsker's) theorem). However, in more general sense the approximation may not always be very accurate, mainly because of the *Join* transitions (those transitions with multiple input places) and the *softened* enabling conditions (a transition is enabled if all of its input places are marked, without considering the weights of arcs). In this chapter, some results related to the approximation by using continuous PNs are described. Moreover, we briefly recall several techniques for improving the approximation (such as introducing *white noise* [101], modifying the server semantics [59]). Other important issues of TCPNs such as *observability* and *fault diagnosis* are not discussed in this chapter, but can be found in, for example, [47, 67, 87, 68].

Among other classical control problems (for instance, *supervisory control*, in which the goal is to design a maximally permissive control to avoid certain forbidden states), we focus on two problems: *target marking (state) control* problem and

optimal flow control problem. The objective is to reach, in *minimum-time*, a desired target (final) state or an optimal flow (obtained in a convex region). The first problem is similar to the classical *set-point* control problem of general continuous-state systems and, the marking of continuous PNs can be viewed as the average marking of the underlying discrete PNs. The final marking, denoted by $\boldsymbol{m}_f$, is usually selected in an early design stage according to some optimality indices, e.g., maximizing the flow in steady states [89]. Since we may not be able to uniquely determine a final state with a given optimal flow, the second problem is usually more complicated, especially when minimum-time evolution is addressed.

Chapter 3 introduces the basic concepts of the control of TCPNs and some fundamental issues, as *controllability*, are recalled. The target marking control of TCPNs, mainly under infinite server semantics, has been discussed in many works (see, for example [36, 64, 84, 45, 51, 102]). In Chapter 3 we summarize some existing control methods, mainly for fully controllable systems. In this work we also assume that all the transitions are *controllable*. Let us point out that, here we focus on the design of controllers based on the continuous models, and we assume that fluidization has been properly done, i.e., the main desirable properties of the original discrete system are preserved in the continuous model. The obtained continuous control laws may be applied back to control the underlying discrete systems, this topic has been discussed in for example, [98]. Fig. 1.1 shows the big picture of the research field and we are interested in the shaded part:

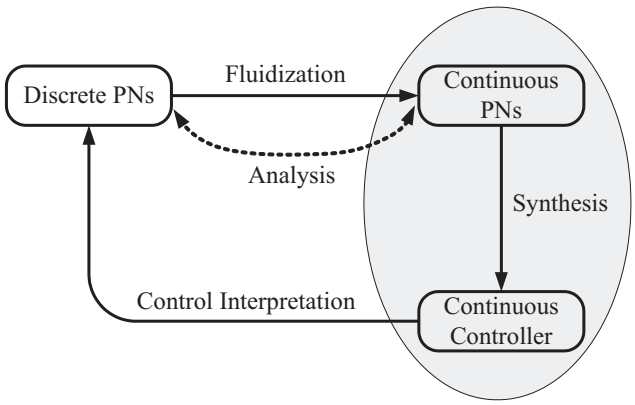


Figure 1.1: The sketch of the research field

Although many works can be found in the literature about the target marking control problem of TCPNs, most of them only focus on the convergence to the final state. From the computational point of view, complexity may grow very quickly (even exponentially) on the size of the net system (for example, the affine control [102], MPC control [64]); meanwhile, very limited works have taken into account some interesting optimality criteria, as minimum-time evolution, in the control synthesis. In Chapter 4, we propose several control methods based on the ON/OFF (or Bang-Bang) strategy, with the objective of driving the system to the final state in

minimum-time. We first propose an ON/OFF controller for Choice-Free (CF) net systems (Section 4.2) and we prove that it ensures a minimum-time time evolution. The idea is rather simple: let all the transitions fire as fast as possible until their upper bounds, given by the *minimal firing count vector* (that can be computed in polynomial time), are reached; then simply block them. Nevertheless, this standard ON/OFF strategy cannot be applied to general net systems because convergence to the final state is not guaranteed; some illustrative examples are given in Section 4.3. Several extensions (ON/OFF+, B-ON/OFF and MPC-ON/OFF) are proposed in Section 4.4, adding more adequate strategies to solve the conflicts that appear in general net systems; all the extended methods are *heuristics* for the minimum-time control. A main advantage of the proposed methods is the low computational complexity; meanwhile, reasonable time spent for reaching the final state can be obtained.

The distributed physical deployment of a large scale system often makes it impossible to implement a centralized controller, considering the high communication costs, time delays, etc. In the context of target marking control problem of TCPNs, few contributions have considered the decentralized setting. For example, [4] considered continuous models composed by several subsystems that communicate through buffers (modelled by places). This method assumes that all the subsystems and the global one should be *mono-T-semiflow*. In Chapter 5 we propose a decentralized control method for CF nets. We assume a large scale system modelled by TCPNs that can be cut through a set of buffer places, obtaining disconnected subsystems. However, these disconnected subsystems may exhibit different behaviors (firing sequences) to the original system. To overcome this problem, we propose several reduction rules to obtain *abstractions* of the missing parts of subsystems. The abstractions are used to construct the *complemented subsystems* that preserve the behaviors of the original system. Then, local control laws are computed separately in subsystems. Finally, we present a simple algorithm to coordinate the local control laws that may be not globally admissible. Because the considered nets are CF, we can implement the ON/OFF controller independently and drive each subsystem to its final state in minimum-time.

For a general net system, the previous decentralized control methods may be no longer applicable: the method proposed in Chapter 5 is only for CF nets; the approach proposed in [4] requires (sub-)systems to be mono-T-semiflow. In Chapter 6 we propose an approach based on *Distributed Model Predictive Control* (DMPC). We first present a centralized MPC controller, in which the stability—a key issue in MPC based approaches—is ensured by forcing the state evolution inside an interior convex subnet of the reachability space. Recall that in another (centralized) MPC control approach for TCPNs proposed in [64], the states are constrained to be on a straight line trajectory from the initial state to the final one; however, for our method this is not mandatory. Later, we apply the proposed MPC controller to a distributed setting. Similarly to the previous methods, we assume a (large scale) TCPN that is cut into subsystems through sets of buffer places. Then we focus on driving all the subsystems to their final states and keeping all the buffer places in legal non-

negative states. In the proposed distributed MPC algorithm, each local controller can access informations (states and structures) of its local subsystem and the buffers connecting to it; no global coordinator is required, and communications among local controllers only occur inside neighborhoods, in which the data transmitted is very low. However, minimum-time evolution is not considered in this method.

In Chapter 7 we are interested in reaching the maximal flow of TCPNs in minimum-time. As we have already mentioned, the main challenge of this problem is the fact that we usually cannot uniquely determine a final state with the maximal flow and obviously, the time varies significantly on which one is chosen. Even for *Marked Graphs* (MG, a subclass of CF nets), the problem becomes complicated when minimum-time evolution is considered, in particular, non-monotonicity appears with respect to the firing count vectors that drive the system to the maximal flow. We propose a heuristic algorithm for CF nets. The idea is to compute the "best" firing counter vector (in terms of the time spent on the trajectory) driving the system to the maximal flow, according to an estimation of the number of time steps based on the current state and flow at each time step; then an ON/OFF firing strategy is applied. Moreover, because of the *persistency* of CF nets, we can further reduce the time by employing some additional firings.

The main contributions of this thesis can be briefly listed as the follows:

- A simple and efficient minimum-time controller for the target marking control problem of CF net systems (Chapter 4, the primary results are published in [104]).
- Several heuristic minimum-time control methods for the target marking control problem of general net systems (Chapter 4, the primary results are published in [106]).
- A decentralized minimum-time controller for the target marking control problem of CF net systems (Chapter 5, the primary results are published in [105, 103]).
- A distributed MPC approach for the target marking control problem of general net systems (Chapter 6, the primary results are in [107])
- Heuristic methods for the minimum-time (maximal) flow control problem of CF nets (Chapter 7, the primary results are published in [108])
- The proposed control methods are implemented and integrated into a Matlab based toolbox for hybrid PN systems, called, SimHPN [48].

The organization of the thesis is as follows: In Chapter 2 we briefly recall the basic concepts and important technical results of continuous PNs; in Chapter 3 more details about the control of continuous PNs, which is the main topic of the thesis, are introduced. In Chapter 4 we propose centralized control methods for the target marking control problem, with the objective of minimizing the time spent

on the trajectory. Some proposed methods are heuristics and all of them are based on the ON/OFF strategy. Chapter 5 and 6 study decentralized control methods for the target marking control problem: in Chapter 5 we propose a decentralized minimum-time controller for CF net systems; in Chapter 6, we propose a distributed MPC approach for general net systems. Chapter 7 focuses on the (minimum-time) optimal flow control problem, and heuristic algorithms are proposed for CF nets. In Chapter 8 we carry out several case studies to illustrate the proposed (target marking) control methods: the first three examples focus on the centralized control methods and the last one considers the distributed control. Some final remarks are in Chapter 9.

Chapter 2

Continuous Petri nets: Basic Concepts and Notations

In this chapter, we introduce some basic definitions, concepts and techniques about continuous Petri nets, both for the autonomous (untimed fully non-deterministic) model and the timed model. Without time interpretation, the autonomous model can be used to analyze some properties like boundedness, deadlock-freeness, liveness, etc. Notice that, as a "coarse" model, some important properties of the original discrete model may be lost after fluidization. Therefore, during the presentation of the technical results related to continuous Petri nets, we will compare with those related to the discrete ones, trying to clarify the "bridges" and "gaps" between them. Timed models are often used in performance evaluation of, for example, manufacturing systems. Among mostly used firing server semantics, we focus on *infinite server semantics* (variable firing speed), since it usually provides better approximations to discrete systems under some general conditions. Finally, we present a case study to illustrate the concepts and techniques that have been introduced in this chapter.

2.1 (Discrete) Petri nets and the state explosion problem

Petri nets are a modelling paradigm with several "related" formalisms. In the sequel, we consider Place/Transition (P/T) nets, which is most usually found in the literature. PNs enjoy a *bipartite* structure, which is also considered in other DEDS formalisms as *queueing networks* or *Forrester Diagrams* (see [87] for a broad review). They can directly represent a *production/consume* logic that frequently appears in practical systems as manufacturing systems, logistics, transportation systems. In this section we introduce the basic definitions of discrete PNs, and illustrate its principal limitation—the state explosion problem.

Definition 2.1.1. *A Petri net (PN) system is a pair* $\langle \mathcal{N}, \boldsymbol{M}_0 \rangle$*, in which* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *is a net structure, where:*

- P *and* T *are the disjoint, finite sets of places and transitions respectively.*
- $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{N}^{|P| \times |T|}$ *are the pre and post incidence matrices.*
- $\boldsymbol{M}_0 \in \mathbb{N}^{|P|}$ *is the initial marking (state).*

Let p_i, $i = 1, \ldots, |P|$ and $t_j, j = 1, \ldots, |T|$ denote the places and transitions. $\boldsymbol{Pre}[p_i, t_j] = w_1$ and $\boldsymbol{Post}[p_i, t_j] = w_2$ indicate the connections between places and transitions: if $w_1 > 0$ there is a arc from p_i to t_j with w_1 as the weight; if $w_2 > 0$ there is a arc from t_j to p_2 with w_2 as the weight. For any $v \in P \cup T$, the sets of its input and output nodes are denoted as ${}^\bullet v$ and $v^\bullet$, respectively. These definitions can be naturally extended to sets of nodes. Each place can contain a non-negative real number of tokens, its marking. The distribution of tokens in places is denoted by $\boldsymbol{M}$ and the marking of place p_i is represented as $\boldsymbol{M}[p_i]$. In (discrete) PNs one transition t_j is enabled at marking $\boldsymbol{M}$ if each of its input place $p_i \in {}^\bullet t_j$ fulfills $\boldsymbol{M}[p_i] \geq \boldsymbol{Pre}[p_i, t_j]$. The enabling degree of transition t_j at marking $\boldsymbol{M}$ is defined as:

$$enab(t_j, \boldsymbol{M}) = \min_{p_i \in {}^\bullet t_j} \left\{ \left\lfloor \frac{\boldsymbol{M}[p_i]}{\boldsymbol{Pre}[p_i, t_j]} \right\rfloor \right\} \tag{2.1}$$

It gives the maximal amount that transition t_j can fire at $\boldsymbol{M}$. Transition t_j is called *k-enabled* under marking $\boldsymbol{M}$, if $enab(t_j, \boldsymbol{M}) = k$. The firing of transition t_j with an amount $\alpha \in \mathbb{N}$ (denoted by $t_j(\alpha)$) leads the system to a new state $\boldsymbol{M'} = \boldsymbol{M}_0 + \alpha \cdot \boldsymbol{C}[P, t_j]$, the evolution being denoted by $\boldsymbol{M} \overset{t_j(\alpha)}{\rightarrow} \boldsymbol{M'}$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the *token flow matrix* (*incidence* matrix if $\mathcal{N}$ is self-loop free) and $\boldsymbol{C}[P, t_j]$ and $\boldsymbol{C}[p_i, T]$ are its j^{th} column and i^{th} row. A marking $\boldsymbol{M}$ that can be reached from $\boldsymbol{M}_0$ by firing a sequence $\sigma = t_1(\alpha_1)t_2(\alpha_2)...$, satisfies the following state (fundamental) equation:

$$\boldsymbol{M} = \boldsymbol{M}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \quad \boldsymbol{M} \in \mathbb{N}^{|P|}, \boldsymbol{\sigma} \in \mathbb{N}^{|T|} \tag{2.2}$$

where $\boldsymbol{\sigma}$ is called the *firing count vector* corresponding to firing sequence σ, such that $\boldsymbol{\sigma}[t_j]$ is the accumulative amount that t_j fires in $\boldsymbol{\sigma}$.

Similar to other modelling formalisms, PNs also suffer from the state explosion problem of DEDS which makes intractable the computational complexity of the traditional state enumeration based methods. In particular, the size of the reachability set of a PN may increase exponentially with respect to the initial state.

Example 2.1.2. *Let consider a discrete net system given in Fig.2.1 [93, 46]. Table 2.1 shows that the size of the reachability set grows exponentially when the initial state is scaled.*

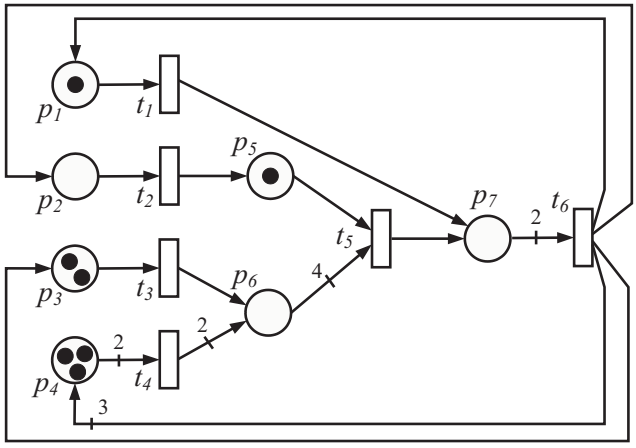


Figure 2.1: A simple PN that models an assembly system, initial state $\boldsymbol{M}_0$ =[1 0 2 3 1 0 0]^T

Table 2.1: The size of the reachability set of the net system in Fig.2.1

Initial state	Size of the reachability set
$\boldsymbol{M}_0$	54
$2 \cdot \boldsymbol{M}_0$	1,685
$3 \cdot \boldsymbol{M}_0$	10,354
$4 \cdot \boldsymbol{M}_0$	37,722
$5 \cdot \boldsymbol{M}_0$	103,914
...	...
$10 \cdot \boldsymbol{M}_0$	2,598,345

2.2 Autonomous (untimed) continuous Petri nets

One classical technique used to overcome the state explosion problem is *fluidization*. Fluid models are obtained by removing the integrality constraint from the system. In particular, in the fluid PN models, the firing of transitions and consequently the

markings, are no longer restricted to the natural and they can be non-negative real numbers. The main advantage of using the fluid relaxation is that the computational issue in the original discrete model is considerably reduced, usually in a dynamical way.

2.2.1 Basic concepts

Definition 2.2.1. *A continuous Petri net (CPN) system is a pair* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *where* $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ *is the same net structure as defined for the discrete PN. The difference is that in CPNs the firing of transitions and the markings (states) are no longer restricted to be in the naturals, but relaxed to be non-negative real numbers, so,* $\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$. *In CPN systems, the markings are denoted by* $\boldsymbol{m}$*, distinguishing with* $\boldsymbol{M}$ *for the markings in discrete models.*

In CPN systems, a transition t is *enabled* at $\boldsymbol{m}$ if for every $p \in {}^\bullet t$, $\boldsymbol{m}[p] > 0$, i.e., every input place should be marked. Notice that, in contrast with discrete systems, it is not necessary to consider the weights of arcs to decide whether a transition is enabled or not. However, the weights of arcs are important to compute the *enabling degree* of a transition t_j at a certain marking $\boldsymbol{m}$, which is defined as:

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

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 $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}[P, t_j]$. Similar to to discrete systems, a reachable marking from $\boldsymbol{m}_0$ through a finite sequence σ is included in the state (fundamental) equation:

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \quad \boldsymbol{m}, \boldsymbol{\sigma} \geq 0 \tag{2.3}$$

2.2.2 Petri nets subclasses

The subclasses of discrete PNs that depend only on the structure of net are also applicable to the continuous PNs; in particular, we consider the following subclasses:

- Marked-Graph(MG) [74]: ordinary net and $\forall p \in P, |p^\bullet| = |{}^\bullet p| = 1$.
- Weighted T-system (WTS) [92]: $\forall p \in P, |p^\bullet| = |{}^\bullet p| = 1$.
- Choice-Free (CF) [93]: $\forall p \in P, |p^\bullet| = 1$.
- Join-Free (JF): $\forall t \in T, |{}^\bullet t| \leq 1$.
- Equal conflict (EQ) [94]: iff ${}^\bullet t \cap {}^\bullet t' \neq \emptyset \Rightarrow \boldsymbol{Pre}[P, t] = \boldsymbol{Pre}[P, t']$.
- Mono-T-semiflow (MTS) [19]: conservative and has a unique minimal T-semiflow whose support contains all the transitions.

2.2.3 Basic structural concepts

The *support* of a vector, $\boldsymbol{v} \geq \mathbf{0}$, is $\|\boldsymbol{v}\| = \{v_i | v_i > 0\}$, the set of positive elements of $\boldsymbol{v}$. Right ($\boldsymbol{C} \cdot \boldsymbol{x} = 0$) and left ($\boldsymbol{y} \cdot \boldsymbol{C} = 0$) natural annullers of the token flow matrix are called T- and P-*semiflows*, respectively. A semiflow is *minimal* when its support is not a proper superset of the support of any other semiflow, and the greatest common divisor of its elements is one. As in discrete nets, when $\exists \boldsymbol{y} > \mathbf{0}$, s.t. $\boldsymbol{y}^T \cdot \boldsymbol{C} = \mathbf{0}$, the net is said to be *conservative*, and when $\exists \boldsymbol{x} > \mathbf{0}$ s.t. $\boldsymbol{C} \cdot \boldsymbol{x} = \mathbf{0}$, the net is said to be *consistent*.

Given a P-semiflow $\boldsymbol{y}$ (a vector), there exist two related notions that should be differentiated:

- *conservation laws*: a set of equations $\boldsymbol{y}^T \cdot \boldsymbol{m}_0 = \boldsymbol{y}^T \cdot \boldsymbol{m}$, which hold for an arbitrary initial marking $\boldsymbol{m}_0$ and every reachable marking $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$, $\boldsymbol{m}, \boldsymbol{\sigma} \geq 0$.

- *conservative component*: the P-subnet generated by the support of $\boldsymbol{y}$. It is a part of the net that conserves its weighted token content.

On the other hand, T-semiflows identify potentially cyclic behaviors in the system, i.e., if $\exists \boldsymbol{x} \gneq \mathbf{0}$ s.t. $\boldsymbol{C} \cdot \boldsymbol{x} = \mathbf{0}$, and $\boldsymbol{x}$ is fireable from $\boldsymbol{m}$ then, by the state equation, $\boldsymbol{m} \xrightarrow{\sigma} \boldsymbol{m}$ with σ being a firing sequence and the corresponding firing count vector is equal to $\boldsymbol{x}$.

Example 2.2.2. *For example, the PN system in Fig. 2.2(a) has a P-semiflow* $\boldsymbol{y} = [1\ 1\ 1]^T$*, therefore* $\|\boldsymbol{y}\| = \{p_1, p_2, p_3\}$*. By the state equation, it holds* $\boldsymbol{y}^T \cdot \boldsymbol{m}_0 = \boldsymbol{y}^T \cdot \boldsymbol{m}$*, i.e., for any marking* $\boldsymbol{m}$ *reachable from a given* $\boldsymbol{m}_0$*,* $\boldsymbol{m}[p_1] + \boldsymbol{m}[p_2] + \boldsymbol{m}[p_3] = \boldsymbol{m}_0[p_1] + \boldsymbol{m}_0[p_2] + \boldsymbol{m}_0[p_3]$*, which are the conservation laws. For example, considering the initial marking* $\boldsymbol{m}_0 = [2\ 0\ 0]^T$ *it holds* $\boldsymbol{m}[p_1] + \boldsymbol{m}[p_2] + \boldsymbol{m}[p_3] = 2$*. The P-subnet generated by* $\|\boldsymbol{y}\|$ *contains all the places of the net, i.e., the whole net is a conservative component. The PN system has also a T-semiflow,* $\boldsymbol{x} = [1\ 1\ 1]^T$*, thus* $\|\boldsymbol{x}\| = \{t_1, t_2, t_3\}$*. Therefore, if every transition fires once, the system returns to the initial marking.*

Two interesting structural concepts are siphons and traps. A set of places Σ is a *siphon* if ${}^\bullet\Sigma \subseteq \Sigma^\bullet$. The dual concept of of siphon, called *trap*, is a set of places Θ such that $\Theta^\bullet \subseteq {}^\bullet\Theta$. An important property is that an empty siphon will remain empty forever; and analogously, in discrete net systems, a marked trap cannot get emptied. Nevertheless, in continuous systems, a trap may be emptied in the limit [83]. For example, p_1 in Fig. 2.2(b) is a trap. But, if we consider the net as continuous, p_1 can be emptied with an infinite firing sequence, see Ex. 2.2.5.

For the PN in Fig. 2.2(a), $\Sigma = \{p_1, p_2\}$ is a siphon since: ${}^\bullet\Sigma = \{t_1, t_3\} \subseteq \{t_1, t_2, t_3\} = \Sigma^\bullet$. Considering the PN in Fig. 2.2(b), $S = \{p_1\}$ is a trap and also a siphon, since $S^\bullet = \{t_1, t_2\} = {}^\bullet S$.

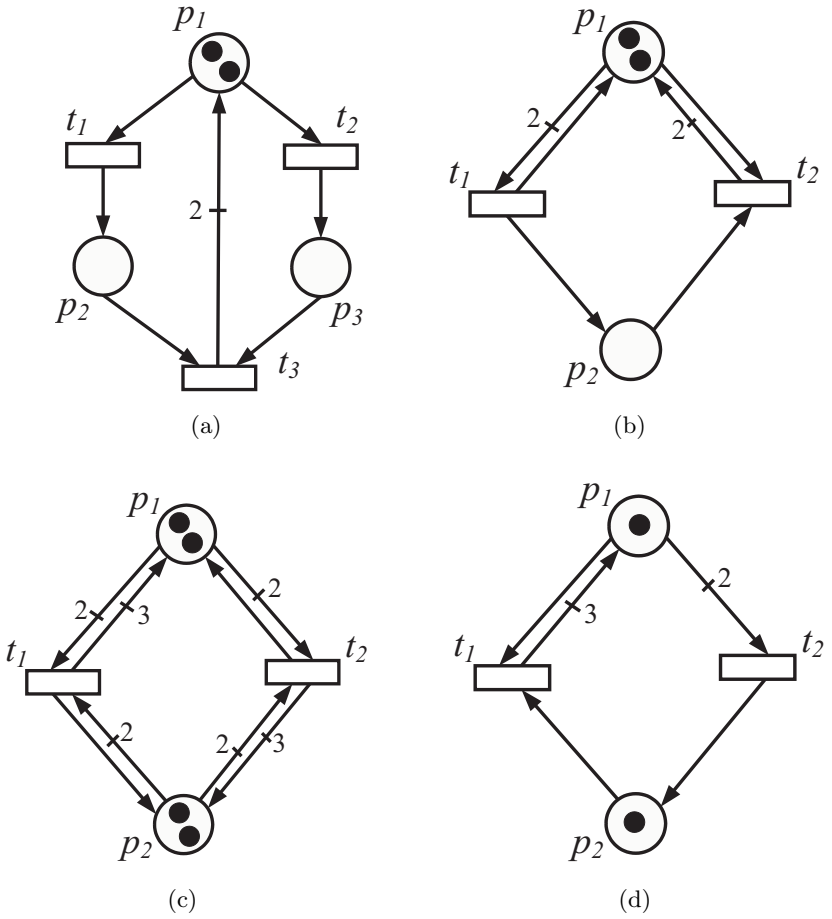


Figure 2.2: Some simple PN systems

2.2.4 Reachability and lim-reachability

The *reachability space* (reachability set) of a given system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, denoted by $\mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$, is the set of all markings that are reachable by a finite firing sequence:

Definition 2.2.3. $\mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0) = \{\, \boldsymbol{m} |$ *a finite fireable sequence* $\sigma = t_{a_1}(\alpha_1) \dots t_{a_k}(\alpha_k)$ *exists such that* $\boldsymbol{m}_0 \overset{t_{a_1}(\alpha_1)}{\rightarrow} \boldsymbol{m}_1 \overset{t_{a_2}(\alpha_2)}{\rightarrow} \boldsymbol{m}_2 \dots \overset{t_{a_k}(\alpha_k)}{\rightarrow} \boldsymbol{m}_k = \boldsymbol{m}$ *where* $t_{a_i} \in T$ *and* $\alpha_i \in \mathbb{R}^+ \}$.

An interesting property of the RS of CPNs, different from the discrete RS is that this set is *convex* [83].

Property 2.2.4. *Let* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *be a continuous PN system. The set* $\mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$ *is convex, i.e., if two markings* $\boldsymbol{m}_1$ *and* $\boldsymbol{m}_2$ *are reachable, then for any* $\alpha \in [0, 1]$, $\boldsymbol{m}' = \alpha \cdot \boldsymbol{m}_1 + (1 - \alpha) \cdot \boldsymbol{m}_2$ *is also a reachable marking.*

Example 2.2.5. *Let us consider the system in Fig. 2.2(b). At the initial marking* $\boldsymbol{m}_0 = [2\ 0]^T$*, transition* t_1 *is enabled, and its enabling degree is 1. It can fire any real amount* α *s.t.* $0 < \alpha \leq 1$*. For example, if it fires the maximal possible amount,* $\alpha = 1$*, the system reaches the marking* $\boldsymbol{m}_1 = [1\ 1]^T$*, from which both transitions (*t_1 *and* t_2*) are enabled. From marking* $\boldsymbol{m}_1$*, if* t_1 *fires an amount equal to* $enab(t_1, \boldsymbol{m}_1) = \frac{1}{2}$*, the system reaches* $\boldsymbol{m}_2 = [\frac{1}{2}\ \frac{3}{2}]^T$*. Firing successively transition* t_1 *an amount equal to its enabling degree, the marking of* p_1 *decreases to the half in each firing; but* p_1 *is never emptied by a finite firing sequence. However, place* p_1 *can be emptied if we consider an infinitely long firing sequence and the marking will approach* $\boldsymbol{m} = [0\ 2]^T$*, which is said to be reachable* in the limit*. Notice that* p_1 *is a trap and it gets emptied in a CPN system, but only with an infinite firing sequence.*

The markings that are reachable with infinite long firing sequences are said to be lim-reachable, denoted by lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$:

Definition 2.2.6. *[83] Let* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *be a continuous system. A marking* $\boldsymbol{m} \in (\mathbb{R}^+ \cup \{0\})^{|P|}$ *is* lim-reachable*, if a sequence of reachable markings* $\{\boldsymbol{m}_i\}_{i \geq 1}$ *exists such that*

$$\boldsymbol{m}_0 \xrightarrow{\sigma_1} \boldsymbol{m}_1 \xrightarrow{\sigma_2} \boldsymbol{m}_2 \cdots \boldsymbol{m}_{i-1} \xrightarrow{\sigma_i} \boldsymbol{m}_i \cdots$$

and $\lim_{i \to \infty} \boldsymbol{m}_i = \boldsymbol{m}$*. The lim-reachable space is the set of lim-reachable markings, and will be denoted by lim-*RS$(\mathcal{N}, \boldsymbol{m}_0)$*.*

For any continuous system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, the differences between RS$(\mathcal{N}, \boldsymbol{m}_0)$ and lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$ are just in the border points of the reachability spaces. Therefore, it holds that RS$(\mathcal{N}, \boldsymbol{m}_0) \subseteq$ lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$ and that the closure of RS$(\mathcal{N}, \boldsymbol{m}_0)$, i.e., all the points in RS$(\mathcal{N}, \boldsymbol{m}_0)$ plus the limit points of RS$(\mathcal{N}, \boldsymbol{m}_0)$, is equal to the closure of lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$ [49]. Moreover, lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$ is also convex.

Assuming an initial marking of non-negative integers of a continuous system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, if $\boldsymbol{m}$ is a marking that is reached by firing transitions in discrete amounts, i.e., as if the system was discrete, then $\boldsymbol{m}$ is also reachable by the system as continuous just by applying the same firing sequence. Thus RS$_D(\mathcal{N}, \boldsymbol{M}_0) \subseteq$ RS$(\mathcal{N}, \boldsymbol{m}_0)$ where $\boldsymbol{M}_0 = \boldsymbol{m}_0$ and RS$_D(\mathcal{N}, \boldsymbol{M}_0)$ is the *discrete* reachability space, i.e., the set of markings reachable in the corresponding discrete system.

Under some common conditions, we can characterize the set lim-RS$(\mathcal{N}, \boldsymbol{m}_0)$ by using some linear inequality systems, which can be easily checked, in polynomial time:

Proposition 2.2.7. *[49, 83] Let* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *be a consistent CPN system, such that each transition can fire at least once (there does not exist an empty siphon at* $\boldsymbol{m}_0$*). Then, the following statements are equivalent:*

- $\boldsymbol{m}$ *is lim-reachable.*
- $\exists \boldsymbol{\sigma} \geq 0$*, such that* $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$*.*
- $\boldsymbol{B}_y \cdot \boldsymbol{m} = \boldsymbol{B}_y \cdot \boldsymbol{m}_0, \boldsymbol{m} \geq 0$*, where* $\boldsymbol{B}_y$ *is a basis of P-flows.*

2.2.5 Boundedness

A PN system is bounded when every place is bounded, i.e., its token content is less than some bounds at every reachable marking. Moreover, it is *structurally bounded* if it is bounded for any initial marking.

By definition, if $\mathcal{N}$ is structurally bounded then $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is bounded, either as a discrete or as a continuous system. Moreover, under general conditions, the opposite is also true for CPNs: if every transition is fireable, i.e. there exists no empty siphon at $\boldsymbol{m}_0$ (a very reasonable condition for real systems), then structural boundedness and boundedness are equivalent.

Property 2.2.8. *[83] Given a CPN system such that every siphon is initially marked, the following statements are equivalent:*

- $\mathcal{N}$ *is structurally bounded*
- $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *is bounded*

The *structural bound* of a place p, $\mathrm{SB}(p)$, in system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ can be computed in polynomial time [90] by solving the following LPP:

$$\begin{array}{ll} max & \boldsymbol{m}[p] \\ s.t. & \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{m}, \boldsymbol{\sigma} \geq 0 \end{array} \tag{2.4}$$

2.2.6 Liveness and deadlock-freeness

Similar to discrete PN systems, liveness and deadlock-freeness of CPNs can be defined as follows:

Definition 2.2.9. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a continuous PN system:*

- $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ deadlocks *if a marking $\boldsymbol{m} \in \mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$ exists such that* $\mathrm{enab}(t, \boldsymbol{m}) = 0$ *for every transition $t \in T$;*
- $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *is* live *if for every transition t and for any marking $\boldsymbol{m} \in \mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$ a successor $\boldsymbol{m}'$ exists such that* $\mathrm{enab}(t, \boldsymbol{m}') > 0$*;*
- $\mathcal{N}$ *is* structurally live *if $\exists\ \boldsymbol{m}_0$ such that $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is live.*

If we consider lim-$\mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$, those concepts are naturally extended to *lim-deadlock*, *lim-live*, and *structurally lim-live*.

We should notice that some properties may be lost when a discrete model is fluidized, in particular, due to the relaxation of the enabling conditions. Thus, the *non-fluidizability* of discrete net systems with respect to deadlock-freeness (also with respect to liveness) may appear, e.g., the new reachable markings might make the system live or might deadlock it. For example, the system in Fig. 2.2(c) deadlocks as discrete after the firing of transition t_1. However, it never gets completely deadlocked

as continuous by a finite firing sequence; the continuous system would only deadlock in the limit. On the other hand, the system in Fig. 2.2(d) is live as discrete but gets blocked as continuous if transition t_2 fires in an amount of 0.5 (the deadlock marking $\boldsymbol{m}_d = [0\ 1.5]^T$ is reached).

An interesting results related to the lim-reachability in continuous nets is that it gives a sufficient condition for liveness of the corresponding discrete one [83]:

Property 2.2.10. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a bounded and lim-live continuous system. Then, $\mathcal{N}$ is structurally live and structurally bounded as discrete net.*

Many techniques have been developed for checking of liveness and deadlock-freeness, since usually they are of those basic requirements in a properly designed system. For instance, *rank theorems* were initially developed for discrete models [82], and later, these results were extended to continuous models for the checking of lim-liveness and boundedness [83], in polynomial time. Rank theorems establish necessary or sufficient conditions for liveness based on consistency, conservativeness and the existence of an upper bound on the rank of the token flow matrix. For continuous EQ systems (and for some other classes of net systems), rank theorems provide a full characterization of lim-liveness and boundedness [83]. Moreover, if the net is not EQ, there exist some transformation rules, namely *equalization* and *release*, to convert non EQ systems into EQ ones [83]; but in this case, only sufficient conditions are available. Another typical method used to investigate the deadlock-freeness is to check the *deadlock-trap* property (DTP) (it holds if every siphon contains an initially marked trap), however it is computationally much more expensive [42].

2.2.7 Implicit places and structurally implicit places

The role of places in PN systems is to constrain the fireability of transitions. An *implicit place* is never the unique to constrain the firing of a transition, thus it could be removed.

Definition 2.2.11. *Given a PN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, the implicit and structurally implicit places can be defined as:*

- *A place p is* implicit *if it is never the unique place that prevents the firing of a transition.*
- *A place p is* structurally implicit *if there exists an initial marking $\boldsymbol{m}_0$ from which it is implicit.*

A characterization of the structurally implicit places is given in [90]:

Property 2.2.12. *Let $\mathcal{N} = \langle P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$. Place p is structurally implicit iff one of the following statements is satisfied (the second is the dual of the first one):*

1. $\exists \boldsymbol{y} \geq \mathbf{0}$, *such that* $\boldsymbol{C}[p,T] \geq \boldsymbol{y} \cdot \boldsymbol{C}[P,T]$ *and* $\boldsymbol{y}[p] = 0$

2. $\nexists \boldsymbol{x} \geq \mathbf{0}$, *such that* $\boldsymbol{C}[P,T] \cdot \boldsymbol{x} \geq \mathbf{0}$ *and* $\boldsymbol{C}[p,T] \cdot \boldsymbol{x} < \mathbf{0}$

Let us remark that, as a necessary condition for a place p to be structurally implicit, it must not be the only input place of its output transitions ($\forall t \in p^\bullet$, $|^\bullet t| \geq 2$). The initial marking from which a structurally implicit place p becomes implicit can be efficiently computed from the initial marking of the rest of the places.

Property 2.2.13. *[90] Let* $\mathcal{N} = \langle P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$. *Place* p *is implicit if* $\boldsymbol{m}_0[p]$ *is greater than or equal to the optimal value of the following linear programming problem (LPP):*

$$\begin{array}{ll} min & \boldsymbol{y} \cdot \boldsymbol{m}_0[P] + \mu \\ s.t. & \boldsymbol{y} \cdot \boldsymbol{C}[P,T] \leq \boldsymbol{C}[p,T] \\ & \boldsymbol{y} \cdot \boldsymbol{Pre}[P,p^\bullet] + \mu \cdot \mathbf{1} \geq \boldsymbol{Pre}[p,p^\bullet] \\ & \boldsymbol{y} \geq 0 \end{array} \tag{2.5}$$

Although implicit places deal only with the redundant information, they are interesting from different points of view: to improve the analysis of the PN (for example, the technique of removing spurious solutions [90, 87]), or to interpret its physical meaning.

In the field of manufacturing systems, an implicit place may represent a kind of resource (robot, machine, or buffer, etc.) that its marking is not constraining the system. Consequently, increasing the number of these resources would not improve the system's throughput.

2.3 Timed continuous Petri nets

2.3.1 Conceptual framework and server semantics

By introducing time to the model, timed PNs are obtained. They are widely used for performance evaluation. A simple and interesting way to introduce time to CPNs is to assume that time is associated to transitions, which is addressed in this thesis. In timed CPNs (TCPNs), the fundamental equation explicitly depends on time: $\boldsymbol{m}(\tau) = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$, 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*, and leads to the following equation for the dynamics of TCPN systems:

$$\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \boldsymbol{f}(\tau). \tag{2.6}$$

Depending on how the flow $\boldsymbol{f}$ is defined, different firing semantics can be obtained, being the most used ones the *finite server* (or constant speed) semantics and *infinite server* (or variable speed) semantics.

Let us assume that a constant firing rate $\boldsymbol{\lambda}[t_j]$ (or denoted by λ_j) is assigned to each transition t_j. For *finite server semantics*, if the markings of the input places of

t_j are strictly greater than zero (*strongly enabled*), its flow will be constant, equal to $\boldsymbol{\lambda}[t_j]$, i.e., all servers work at the maximal speed. Otherwise (*weakly enabled*), the flow of t_j will be the minimum between its maximal firing speed and the total input flow to the empty places (hence, $\boldsymbol{\lambda}[t_j]$ represents the product of the number of servers in the transition and their speed). This corresponds to the *constant speed* of [1], where the flow of a transition t_j is:

$$\boldsymbol{f}[t_j] = \begin{cases} \boldsymbol{\lambda}[t_j], \text{ if } \forall p_i \in {}^\bullet t_j, m_i > 0 \\ \min\left\{ \min\limits_{p_i \in {}^\bullet t_j | m_i = 0} \left\{ \sum\limits_{t_q \in {}^\bullet p_i} \frac{\boldsymbol{f}[t_q] \cdot \boldsymbol{Post}[t_q, p_i]}{\boldsymbol{Pre}[p_i, t_j]} \right\}, \boldsymbol{\lambda}[t_j] \right\}, \qquad \text{otherwise} \end{cases} \tag{2.7}$$

The dynamical system under finite servers semantics corresponds to a *piecewise constant* system; a switch occurs when the set of empty places changes and the new flow values must ensure that the marking of all places remains positive.

In this thesis we focus on the *infinite server semantics*. The flow of transition t_j is given by:

$$\boldsymbol{f}[t_j] = \boldsymbol{\lambda}[t_j] \cdot enab(t_j, \boldsymbol{m}) = \boldsymbol{\lambda}[t_j] \cdot \min_{p_i \in {}^\bullet t_j} \left\{ \frac{\boldsymbol{m}[p_i]}{\boldsymbol{Pre}[p_i, t_j]} \right\}, \tag{2.8}$$

In TCPNs under *infinite server semantics*, the flow through a transition t_j is the product of its firing rate and its enabling degree. Due to the existence of *minimum* operator, the dynamical system corresponds to a *piecewise linear* system and it induces several strongly related concepts:

a) the set of arcs (p, t), one per transition $t \in T$, in which $p \in P$ is the place defining the enabling degree of t at marking $\boldsymbol{m}$, is known as *configuration* at $\boldsymbol{m}$;

b) the sub-state space in which the configuration is the same is known as *region*;

c) at each region the dynamics is driven by a single *linear system* which is also known as *operation mode*.

More formally:

Definition 2.3.1. *A* configuration *of a net $\mathcal{N}$ is a set of (p, t) arcs, one per transition,* covering *the set T of transitions. Associated to a given configuration $\mathcal{C}_k$ is the following $|T| \times |P|$ configuration matrix:*

$$\boldsymbol{\Pi}_k[t, p] = \begin{cases} \frac{1}{\boldsymbol{Pre}[p,t]}, & \text{if } (p, t) \in \mathcal{C}_k \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

Definition 2.3.2. *A* region *of a net system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is a subset of the reachability space, denoted by $\mathcal{R}_i(\mathcal{N}, \boldsymbol{m}_0) \subseteq \mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$, such that for any two states $\boldsymbol{m}_a, \boldsymbol{m}_b \in \mathcal{R}_i(\mathcal{N}, \boldsymbol{m}_0)$, the corresponding configuration matrices are the the same, i.e., $\boldsymbol{\Pi}(\boldsymbol{m}_a) = \boldsymbol{\Pi}(\boldsymbol{m}_b)$.*

Let us notice that regions are disjoint except on the borders and the reachability set $\mathrm{RS}(\mathcal{N}, \boldsymbol{m}_0)$ of a TCPN system can be partitioned according to the configurations and inside each obtained *convex region* $\mathcal{R}_i(\mathcal{N}, \boldsymbol{m}_0)$ the system dynamic is linear. According to (2.6), (2.8) and (2.9), the dynamic system evolution inside a region $\mathcal{R}_k$, called *operation mode* k as well, can be written as:

$$\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \boldsymbol{f}(\tau) = \boldsymbol{C} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{\Pi}(\boldsymbol{m}) \cdot \boldsymbol{m}(\tau), \tag{2.10}$$

where $\boldsymbol{\Lambda} = diag(\boldsymbol{\lambda})$ is a diagonal $|T| \times |T|$ matrix containing the firing rates of transitions and $\boldsymbol{\Pi}(\boldsymbol{m}) = \boldsymbol{\Pi}_k$ is the configuration matrix associated to $\mathcal{R}_k$ (if $\boldsymbol{m}$ is on the border of two regions $\mathcal{R}_1$ and $\mathcal{R}_2$, any operation mode with $\boldsymbol{C\Lambda\Pi}_1$ or $\boldsymbol{C\Lambda\Pi}_2$ can be used since the same behavior is obtained). The number of modes (regions, configurations) is upper bounded by $\prod_{t \in |T|} |{}^\bullet t|$ but some of them may be redundant and can be removed [67].

It has been proved that TCPNs, under infinite server semantics, have the capability to simulate Turing machines [79], thus they have an important expressive power; nevertheless, certain properties are *undecidable* (for example, marking coverability, submarking reachability or the existence of a steady-state).

There also exist other server semantics. For instance, in population systems (predator/prey, biochemistry, ...), the transition firing flows are usually described by *products* of markings (population semantics), and even more specific non-linear functions (see, for example, [88, 35]). In fact, the products can be obtained from infinite server semantics while considering discoloration of colored PN models [88]. From a different perspective, an extension of the infinite server semantics is defined in [37] where lower and upper bounds are given for the firing rates. The idea is that using *interval firing speeds* the variability of the stochastic behavior of the underlying discrete model can be taken into account in performance evaluation tasks.

Among other semantics, the *finite server semantics* and *infinite server semantics* are mainly used, for example, in manufacturing or logistic systems. In [25], the authors observed that most frequently the infinite server semantics approximates better the marking of the discrete net system. Moreover, for *mono-T-semiflow reducible* net systems [50] under some general conditions it is proved that infinite server semantics approximates better the flow in steady state [66]. The result holds depending on a structural property defined from the steady-state marking, a condition that is quite common in the case of production systems. In the sequel, we assume TCPNs under infinite server semantics.

2.3.2 Timed models versus untimed models

Assume that the *steady-state* exists, and let $\boldsymbol{f}_{ss}$ be the flow vector of the timed system in the steady state, $\boldsymbol{f}_{ss} = \lim_{\tau \to \infty} \boldsymbol{f}(\tau)$, from (2.6) $\dot{\boldsymbol{m}} = \boldsymbol{C} \cdot \boldsymbol{f}_{ss} = 0$ is obtained (independently of the firing semantics, the flow in the steady state is a T-semiflow of the net). Deadlock-freeness and liveness definitions of untimed systems can be easily extended to timed systems as follows:

Definition 2.3.3. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a timed continuous PN system and $\boldsymbol{f}_{ss}$ be the vector of flows of the transitions in the steady state.*

- $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *is* timed-deadlock-free *if* $\boldsymbol{f}_{ss} \neq \mathbf{0}$*;*
- $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *is* timed-live *if* $\boldsymbol{f}_{ss} > \mathbf{0}$*;*
- $\langle \mathcal{N}, \boldsymbol{\lambda} \rangle$ *is* structurally timed-live *if* $\exists\ \boldsymbol{m}_0$ *such that* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *is* timed-live*.*

The relationships among liveness of timed systems under infinite server semantics and untimed systems are depicted in Fig. 2.3. When we associate time to the system, we just give a particular trajectory of the untimed system. Thus, there exists a one way bridge between the (structurally) lim-liveness and (structurally) timed-liveness: the lim-liveness (lim-deadlock-free) in an untimed system implies timed-liveness (timed-deadlock-free) of the system if it is considered as timed, but the reverse is not true. On the other hand, if the untimed system is non-live, particular numerical timings of the continuous model can eventually transform it into live. For example, the system Fig. 2.2(a) deadlocks as untimed but is timed-live with $\boldsymbol{\lambda}$ =[1 1 2]^T (in particular $\boldsymbol{f}_{ss}$ = [1 1 1]^T). The results hold even for deadlock-free marking non-monotonic systems (i.e., systems that being deadlock-free, run into a deadlock if the initial marking is increased). More results related to the time-dependent liveness of TCPNs under infinite server semantics can be found in [100]

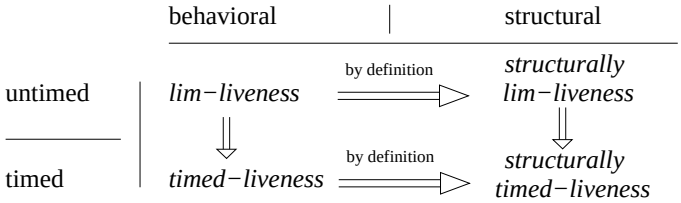


Figure 2.3: Relationships among liveness definitions for continuous models [87]

2.3.3 Performance bounds under infinite server semantics

The throughput of a transition in the steady state (if exists), i.e., the number of firings per time unit, is an important performance index in the evaluations of systems modelled as discrete PNs. In the continuous approximation, this corresponds to the firing flow in steady state. Let us consider MTS [50] which represents an important generalization of CF nets [93] and has reasonable modelling powers.

Assume that the system is consistent and does not have an empty siphons at $\boldsymbol{m}_0$, then from Proposition 2.2.7, every lim-reachable marking is included in the state equation; on the other hand since in MTS there exists a unique minimal T-semiflow that includes all its transitions, the throughput (flow) of system can be

computed using the following *non*-linear programming problem that maximizes the flow of an arbitrary transition t_j:

$$\begin{array}{ll} \max & \boldsymbol{f}_{ss}[t_j] \\ \text{s.t.} & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{f}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \min\limits_{p_i \in {}^\bullet t} \left\{ \frac{\boldsymbol{m}_{ss}[p_i]}{\boldsymbol{Pre}[p_i,t_j]} \right\}, \forall t \in T \\ & \boldsymbol{C} \cdot \boldsymbol{f}_{ss} = 0 \\ & \boldsymbol{m}_{ss}, \boldsymbol{\sigma} \geq 0 \end{array} \tag{2.11}$$

where $\boldsymbol{m}_{ss}$ is the steady-state marking. Due to the *minimum* operator, problem (2.11) is non linear and a branch & bound algorithm was propsed in [50] to solve it. By relaxing the minimum operator to inequalities the problem is reduced to a LPP, shown in (2.12), which can be solved in polynomial time, but usually we may only obtain a non-tight upper bound, i.e., the solution may be not reachable if there exists a transition for which the flow equation is not satisfied. If the net is not MTS, similar developments can be done by adapting the equations in [23].

$$\begin{array}{ll} \max & \boldsymbol{f}_{ss}[t_j] \\ s.t. & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{f}_{ss}[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_S, \forall p \in {}^\bullet t \\ & \boldsymbol{f}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_U, p = {}^\bullet t \\ & \boldsymbol{C} \cdot \boldsymbol{f}_{ss} = \boldsymbol{0} \\ & \boldsymbol{m}_{ss}, \boldsymbol{\sigma} \geq \boldsymbol{0} \end{array} \tag{2.12}$$

where T_U is the set of transitions with *unique* input place, and T_S the *synchronizations* transitions ($T_U \cap T_S = \emptyset$, $T_U \cup T_S = T$) .

Once a solution of LPP (2.12) is obtained, it can be easily checked whether it is the exact value of the flow by introducing it into the problem (2.11).

2.3.4 Parametric optimization under infinite server semantics

Parametric optimization considers "off line" problems in which, given the system configuration, it is optimally parameterized for the steady state.

Among the problems belonging to parametric optimization, some of them are, for example, computing the *optimal* initial marking $\boldsymbol{m}_0$ to achieve the maximal throughput in the steady state, satisfying certain constraints; or problems of minimizing certain cost function related to the initial marking; or optimizing other design parameters, like the *optimal routing* or the *optimal firing speed*, etc.

A general formulation for this class of optimization problems with respect to the steady state is trying to *maximize a profit function* depending on the throughput (flow) vector ($\boldsymbol{f}_{ss}$) in the steady state, the marking in the steady state ($\boldsymbol{m}_{ss}$), and the initial marking ($\boldsymbol{m}_0$). The profit function can be represented, in linear terms, like: $\boldsymbol{g} \cdot \boldsymbol{f}_{ss} - \boldsymbol{w} \cdot \boldsymbol{m}_{ss} - \boldsymbol{b} \cdot \boldsymbol{m}_0$, where $\boldsymbol{g}$ is a gain vector w.r.t. the flow; $\boldsymbol{w}$ is the cost vector due to immobilization to maintain the production flow, e.g. due to the

levels in stores; and vector $\boldsymbol{b}$ represents depreciations or amortization of the initial investments w.r.t. $\boldsymbol{m}_0$, e.g., the size of buffers, the number of machines.

Given $\boldsymbol{K} \cdot \boldsymbol{m}_0 \leq \boldsymbol{d}$ as linear cost-constraints to the initial state, assume that we need to optimize the throughput of transition t_j in the steady state, $\boldsymbol{f}_{ss}[t_j]$, the following LPP can be written [89]:

$$\begin{array}{ll} \max & \boldsymbol{f}_{ss}[t_j] \\ s.t. & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{f}_{ss}[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_S, \forall p \in {}^\bullet t \\ & \boldsymbol{f}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_U, p = {}^\bullet t \\ & \boldsymbol{C} \cdot \boldsymbol{f}_{ss} = \boldsymbol{0} \\ & \boldsymbol{\sigma}, \boldsymbol{m}_0, \boldsymbol{m}_{ss} \geq \boldsymbol{0} \\ & \boldsymbol{K} \cdot \boldsymbol{m}_0 \leq \boldsymbol{d} \end{array} \tag{2.13}$$

where T_U is the set of transitions with unique input place, and T_S the remaining (synchronization) transitions.

If we compare LPP (2.13) with the LPP (2.12), the only difference is that now the initial state $\boldsymbol{m}_0$ appears as a variable and that the linear cost-constraints associated to $\boldsymbol{m}_0$ are added. In general, LPP (2.13) just provides an upper bound of the throughput of transition t_j.

Another parametric optimization problem concerns computing the minimal cost initial marking w.r.t. a given cost weight vector $\boldsymbol{b}$ such that a certain cycle time $\Gamma = 1/\boldsymbol{f}_{ss}[t_j]$ is guaranteed. This optimization problem can be solved by means of the following LPP [89]:

$$\begin{array}{ll} \min & \boldsymbol{b} \cdot \boldsymbol{m}_0 \\ s.t. & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{f}_{ss}[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_S, \forall p \in {}^\bullet t \\ & \boldsymbol{f}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p]}{\boldsymbol{Pre}[p,t]}, \forall t \in T_U, p = {}^\bullet t \\ & \boldsymbol{C} \cdot \boldsymbol{f}_{ss} = \boldsymbol{0} \\ & \boldsymbol{\sigma}, \boldsymbol{m}_0, \boldsymbol{m}_{ss} \geq \boldsymbol{0} \\ & \boldsymbol{f}_{ss}[t_j] \geq 1/\Gamma \end{array} \tag{2.14}$$

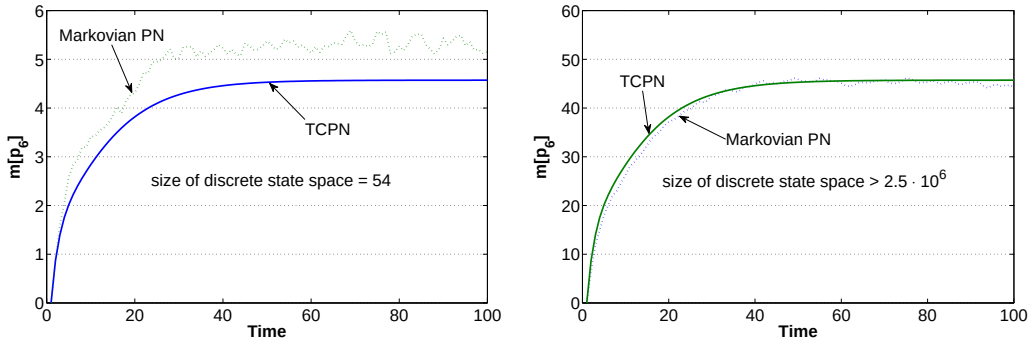
where T_U is the set of transitions with *unique* input place, and T_S the *synchronizations* transitions ($T_U \cap T_S = \emptyset$, $T_U \cup T_S = T$) .

2.3.5 Approximation to the discrete systems

The fluid PNs are a relaxation/approximation of the original discrete model, in particular, we consider the Markovian (discrete) Petri nets (MPNs): stochastic discrete PNs with exponential delays associated to the transitions and conflicts solved by a race policy [73]. MPNs enjoy the memoryless property, and it is widely used in the performance evaluation, for example [73, 69, 8]; but the analysis of its underlying Markovian Chain may be intractable, because of the computational issue cased by

the state explosion problem. The approximation (steady-state as well as transient behavior) of MPNs by using TCPNs under infinite server semantics was first considered in [80]. However, in some situations the fluid approximation may not be good. Therefore it is interesting to investigate the conditions, based on which an appropriate fluid model could be obtained.

Example 2.3.4. *Let us still consider the net system in Fig.2.1. We simulate it by using the Markovian PN model [73] and the corresponding TCPN model under infinite server semantics. The state trajectories of both cases are illustrated in Fig. 2.4. We can see that the fluid model has a reasonable approximation of the original discrete one, and the accuracy is improved if the system is more populated. Notice that, if the initial marking is increased form* $\boldsymbol{m}_0$ *to* $10 \cdot \boldsymbol{m}_0$*, the size of the state space of the discrete PN model increases from 54 to more than 2.5 million—the state explosion problem appears, so the analysis based on the discrete model could be difficult. However, if we consider the fluid model, the number of variables in the system, determined by the number of places, is not changed. On the other hand, for the analysis by using the deterministic fluid model, only one round simulation is enough, which is also much cheaper than using the (stochastic) discrete model.*



(a) Simulation results using initial state $\boldsymbol{m}_0$ (b) Simulation results using initial state $10 \cdot \boldsymbol{m}_0$

Figure 2.4: Simulations: a discrete PN and its fluid model

There are two main reasons that may introduce errors to the fluid models: the *weights* on arcs and *join* transitions (rendez-vous). Let $\boldsymbol{M}$ be the marking of the original discrete PN and $\boldsymbol{m}$ be the one of the corresponding fluid model, TCPN. We assume that the state of fluid model approximates the one of discrete model, then we have $\boldsymbol{m} \sim E(\boldsymbol{M})$, where $E(\boldsymbol{M})$ refers to the *expectation* of $\boldsymbol{M}$. Assume a JF nets, and w be the weight on the directed arc from place p_i to t_j. The expected enabling degree of t_j in the discrete model is $E(enab(t_j, \boldsymbol{M})) = E(\lfloor \boldsymbol{M}[p_i]/w \rfloor)$; while in the TCPN, $enab(t_j, \boldsymbol{m}) = \boldsymbol{m}[p_i]/w \sim E(\boldsymbol{M}[p_i])/w$. Clearly, due to the operation $\lfloor \cdot \rfloor$, $E(enab(t_j, \boldsymbol{M}))$ may be different to $enab(t_j, \boldsymbol{m})$ in a non-ordinary net ($w > 1$). The similar problem may appear even in an ordinary net when t_j is a join ($|{}^\bullet t_j| > 1$): $E(enab(t_j, \boldsymbol{M}) = E(\min\{\boldsymbol{M}[p_i]\}), p_i \in {}^\bullet t_j$ is not equal to

$enab(t_j, \boldsymbol{m}) = \min\{\boldsymbol{m}[p_i]\} \sim \min\{E(\boldsymbol{M}[p_i])\}, p_i \in {}^\bullet t_j$, because it is a common knowledge that operator *min* and E cannot commute. More detailed explanations and illustrative examples about these issues can be found in [87].

It has been formally proved in [101] that for ordinary JF nets, perfect approximation of the discrete model can be obtained by using TCPNs. If a JF net is not ordinary, approximation errors may appears; however, if the net system has a unique asymptotically stable equilibrium point, the errors are ultimately bounded and the larger the average enabling degree the lower the errors. For non JF nets, if the probability that the MPN system evolves inside a unique region (in which the TCPN also evolves) is near 1, i.e., for each synchronization, it is almost always constrained by a single input place, the approximation error is also ultimately bounded and can be improved if the average enabling degree is larger. In [29], the conditions for an appropriate fuidization are investigated mainly based on the *marking homothetic behaviours* of the system. In particular, the relations between the original discrete model and the fluid one are established, in terms of some important logical properties as boundedness, deadlock-freeness, liveness and reversibility.

Several techniques have been proposed to improve the approximation of using TCPNs. For instance, by adding *white noise* to the flows of transitions of the TCPN model [101], a *continuous stochastic* CPN (denoted by TnCPN) is obtained. Intuitively, the stochastic behavior of the MPN is better approximated, according to the following evolution (in discrete time): $\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \boldsymbol{C}(\boldsymbol{\Lambda\Pi}(\boldsymbol{m}_k)\boldsymbol{m}_k\Delta\tau + \boldsymbol{v}_k)$ where $\boldsymbol{v}_k$ is a noise column vector, of length $|T|$, whose elements are of independent normally distributed random variables with zero mean and covariance matrix. An interesting issue is that, by adding the white noise according to the previous approach, the expected value and covariance of the original MPN and the resulting TnCPN coincide.

Another class of techniques for improving the approximation consists in modifying the server semantics of TCPNs. For example, we may change the infinite server semantics by multiplying a marking-dependent function $(\boldsymbol{m}[p_i]^{q-1}/q^q)$ to the flow of a transition t_j, such that $p_i = {}^\bullet t_j$ and $\boldsymbol{Pre}[p_i, t_j] = q$ [87], then the flow of t_j is modified to $\boldsymbol{f}[t_j] = \lambda_j \cdot (\boldsymbol{m}[p_i]/q)^q$. In this way, the approximation may be improved. Belonging to the same category, in [59], the firing rate is considered as piece-wise constant, depending on the regions of the current markings. It is show that the asymptotic mean marking of discrete model can be approximated by the continuous one, if the system is in *non-critical* regions (each join is driven by different place); in [56] the case of critical regions is considered, by means of partial homothetic initial markings but differently, the firing rate is not piece-wise constant but fixed value. More constructive method has been proposed by the same authors in [57], where the homothetic approach is used to compute a set of reference data for several firing rates and an interpolation method is applied.

We can also improve the approximation by removing spurious solutions: those markings that cannot be reached in the discrete model but become (lim-)reachable after fluidization (it is an immediate result of Proposition 2.2.7). Spurious solutions may appear due to the fact that in TCPNs, marked traps are finally emptied in

the limit. Fortunately, this kind of spurious solutions can be *cut* by adding some implicit places to the system. A comprehensive discussion of this technique can be found in [87].

In this thesis, we focus on the synthesis of controllers directly based on the fluid model and we assume that the approximation of the TCPN to the underlying discrete model is appropriate.

2.4 An example: a kanban-like manufacturing system

This section is devoted to illustrate some of the basic concepts and techniques about TCPNs that we have introduced in this chapter, by means of the analysis of the model of a flexible manufacturing system.

The system is composed by two production lines with three machines M1, M2 and M3. The layout of the system and its production process are shown in Fig.2.5, while the PN model is depicted in Fig. 2.6. Parts of type A are processed in machine M1 and then in machine M2, with intermediate products stored in buffers B_1A and B_2A. Parts of type B are first processed in M2 then in M1, with intermediate products stored in buffers B_1B and B_2B. Finally, machine M3 assembles a part A and a part B, obtaining the final product that is stored in buffer B_3 until its removal. Places Max_B_1A and Max_B_1B initially have only one token, so there can be at most one part of type A and one part of type B either in B_1A and B_1B, or being processed by M1 and M2. Parts A and B are moved in pallets all along the process, and there are 20 pallets of type A and 15 pallets of type B. Place Max_B_3 has initially one token, so only one final product can be stored in the buffer B_3 until its removal. The initial state $\boldsymbol{m}_0$ of the system is as shown in Fig. 2.6.

Typical *competition* and *cooperation* relationships that often appear in manufacturing systems, are introduced by means of the movement of parts inside the system. For instance, machine M1 and machine M2 are shared for processing parts A and parts B, therefore, these activities are in mutual exclusion (*mutex*). Final products can be assembled only when both intermediate produces of type A and B are available (i.e., buffer B_2A and B_2B are not empty) (*rendez-vous*).

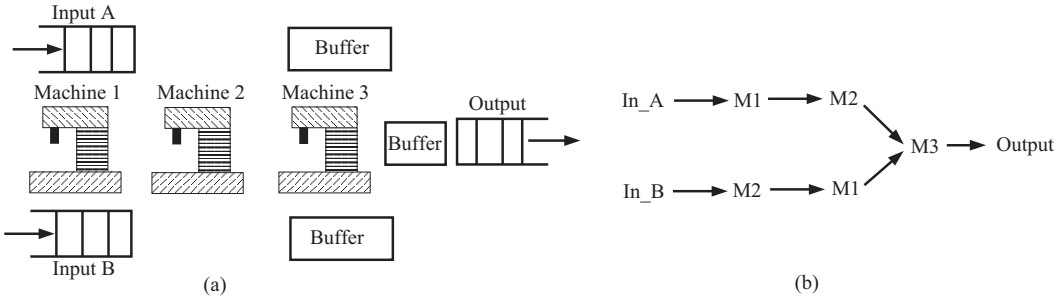


Figure 2.5: (a) Logical layout of a manufacturing system and (b), its production process

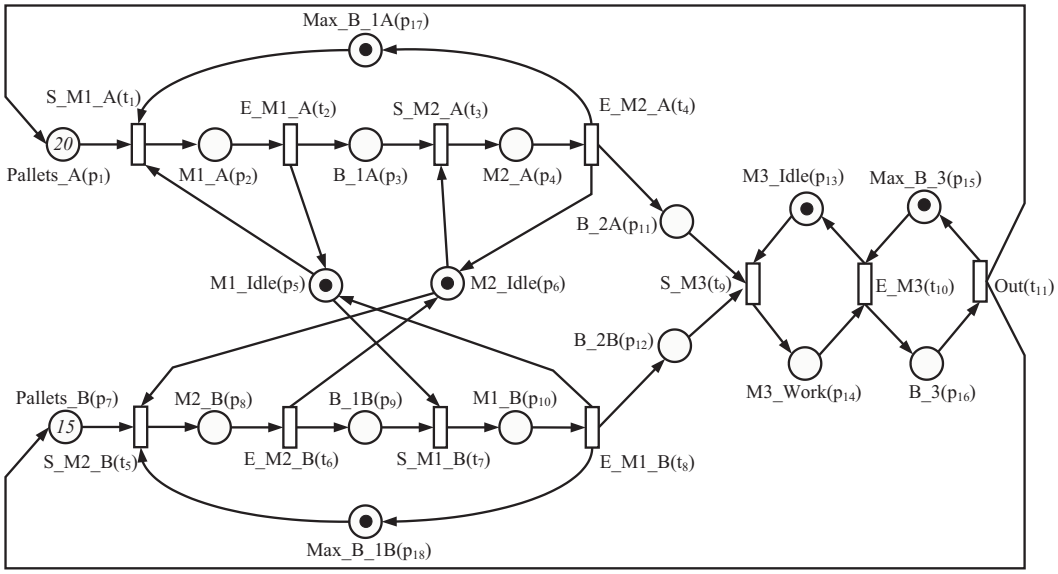


Figure 2.6: The PN system that models the manufacturing system described in Fig.2.5

Let us first consider the net system in Figure 2.6 as a continuous model without any temporal interpretation (i.e., as an autonomous PN: a fully *non-deterministic* model). Some important properties of the autonomous PN system can be studied: conservativeness/consistency, boundedness, structurally implicit places, deadlock-freeness, liveness.

Looking at the structure of the net, it can be checked that it is *conservative*: $\exists \boldsymbol{y} > 0$ s.t. $\boldsymbol{y}^T \cdot \boldsymbol{C} = 0$ (it has 8 elementary P-semiflows covering all the places, each one gives an elementary token conservation law (see Table 2.2)). The system is also *consistent*: it has a unique minimal T-semiflow $\boldsymbol{x} = \mathbf{1}$ ($\boldsymbol{C} \cdot \boldsymbol{x} = 0$). Given that the net is conservative and has a unique T-semiflow that covers all the transitions, the net system is mono-T-semiflow (MTS).

Table 2.2: P-semiflow of the system in Fig. 2.6

P-semiflow	Corresponding token conservation law
$\boldsymbol{y}_1$	M1_A + M1_Idle + M1_B = 1
$\boldsymbol{y}_2$	Pallets_B + M2_B + B1_B + M1_B + B_2B + M3_work + B3 = 15
$\boldsymbol{y}_3$	M2_B + B1_B + M1_B + Max_B1_B = 1
$\boldsymbol{y}_4$	Pallets_A + M1_A + B1_A + M2_A + B_2A + M3_work + B3 = 20
$\boldsymbol{y}_5$	M1_A + B1_A + M2_A + Max_B1_A = 1
$\boldsymbol{y}_6$	M2_A + M2_Idle + M2_B = 1
$\boldsymbol{y}_7$	Max_B_3 + B_3 = 1
$\boldsymbol{y}_8$	M3_Idle + M3_work = 1

(Structural) Boundedness: The PN is conservative, thus it is structurally bounded (i.e., bounded for any $\boldsymbol{m}_0$). The structural bound of each place can be computed from (2.4). For example, $\mathrm{SB}(p_1) = 20$ and $\mathrm{SB}(p_2) = 1$.

Structurally implicit places: There exist six structurally implicit places (see Proposition 2.2.12): M1_Idle, M2_Idle, M3_Idle, Max_B_3, Max_B_1A and Max_B_1B. The minimal initial marking of p_5 to make it implicit (see Proposition 2.2.13) is $\boldsymbol{m}_0'[p_5] = 2$. It means that, if we keep the initial marking of other places and we have 2 or more tokens in p_5, then it will no longer restrict the system. In other words, even we put more machines of type M1 in the system, the throughput cannot be improved. Analogously, the minimal initial marking of p_6 to become implicit is 2; for p_{13}, is 15; for p_{15}, is 15; for p_{17}, is 20; and finally for p_{18}, it is 15.

Deadlock-freeness and liveness: As we have briefly recalled, there exist several methods that can be used for checking the deadlock-freeness. For instance, we can use the deadlock-trap property (DTP). In this particular ordinary net example, siphons are also traps (see Table 2.3) and they are initially marked. So, every siphon contains a marked trap, i.e., the DTP property holds. Thus the net system is deadlock-free. Moreover, the DTP property guarantees not only homothetic deadlock-freeness, but also monotonic deadlock-freeness. It means that, if the marking of any place is increased, the net system will remain deadlock-free. If a discrete system $\langle \mathcal{N}, \boldsymbol{M}_0 \rangle$ is homothetic DF, then it is also DF as continuous [29]. Another interesting way to approach the problem is the following: places $M1\_Idle(p_5)$ and $M2\_Idle(p_6)$ are structurally implicit, if we add enough tokens to the initial state of those places (one more token to each of them), they become implicit. Therefore both can be removed without affecting structural liveness. After the removal, the remaining PN is a strongly connected marked graph with all circuits (i.e., P-semiflows) marked. Thus, the original system is structurally live.

Table 2.3: Minimal siphons of the net. They coincide with the minimal traps.

Minimal siphons / minimal traps
$\{p_1, p_2, p_3, p_4, p_{11}, p_{14}, p_{16}\}$
$\{p_2, p_3, p_4, p_{17}\}$
$\{p_2, p_5, p_{10}\}$
$\{p_4, p_6, p_8\}$
$\{p_7, p_8, p_9, p_{10}, p_{12}, p_{14}\}$
$\{p_8, p_9, p_{10}, p_{18}\}$
$\{p_{13}, p_{14}\}$
$\{p_{15}, p_{16}\}$

Let us now consider the model in Fig.2.6 as a timed PN system. We assume that each transition is associated to a *time delay* that follows exponential distributions. In particular, the time delay vector of transitions, represented by $\boldsymbol{\delta}$, is set as following.

The transitions that model the starting of machines (labelled by S) have time delays $\boldsymbol{\delta}[t_1] = \boldsymbol{\delta}[t_3] = \boldsymbol{\delta}[t_5] = \boldsymbol{\delta}[t_7] = \boldsymbol{\delta}[t_9] = 1$ t.u.. The delays of transitions that model the endings (labelled by E) are $\boldsymbol{\delta}[t_2] = \boldsymbol{\delta}[t_6] = 3$ t.u., $\boldsymbol{\delta}[t_4] = \boldsymbol{\delta}[t_{10}] = 4$ t.u. and $\boldsymbol{\delta}[t_8] = 5$ t.u.. The output transition has a delay $\boldsymbol{\delta}[t_{11}] = 1$ t.u. In the corresponding TCPN model under infinite server semantics, time delays are approximated by their mean values ($\boldsymbol{\lambda}[t_j] = 1/\boldsymbol{\delta}[t_j]$, $t_j \in T$), obtaining a first order (or deterministic) relaxation of the discrete case [80].

As an important part in the life-cycle of manufacturing systems, performance evaluation has been widely investigated by using time interpreted PNs, under both the framework of continuous and discrete systems, for example in [50, 69, 8]. We will focus on the *steady state* evaluation and *transient state* evaluation, using TCPNs under infinite server semantics.

Steady state analysis: By solving LPP (2.12), an upper bound of the flow, equal to 0.1, is obtained (given that the net is MTS with the unique minimal T-semiflow equal to $\mathbf{1}$, all the transitions will have the same flow in the steady state). We can check that it is a solution of problem (2.11), i.e., the relaxed LPP (2.12) gives the exact upper bound of the flow! On the other hand, if we consider the problem (2.11) with *min* operator in the objective function, instead of *max* operator, i.e., computing the lower bound of the flow, the obtained flow is also 0.1. The direct consequence is that, the flow of transitions is exactly equal to 0.1 in the steady state, which is $\boldsymbol{m}_{ss} = [0.1\ 0.3\ 0.1\ 0.4\ 0.2\ 0.3\ 13.5\ 0.3\ 0.1\ 0.5\ 18.6\ 0.1\ 0.6\ 0.4\ 0.9\ 0.1\ 0.2\ 0.1]^T$.

Transient analysis: The transient evaluation analyzes the behavior of the system, from the initial state (at time zero) until a given end time. As we have mentioned in the previous sections, TCPN models can approximate the average marking of the corresponding MPN if the MPN evolves inside a unique region (in which the TCPN also evolves), but it does not hold for this net system. In Fig.2.7 the transient state evolution of M1_Idle (p_5) is shown (obtained with the initial marking shown in Fig. 2.6). The results of the MPN are obtained by 100 simulations and taking the average value at each time instant. It can be observed that, even if the general shape of curves of the TCPN and the MPN are similar, the approximation provided by the fluid model is not very accurate: in the interval from 3.5 t.u. to 4.5 t.u. the average value of $\boldsymbol{M}[p_5]$ (for the MPN) is 0.40, while the average value of $\boldsymbol{m}[p_5]$ (for the TCPN) is 0.22, with error of $(0.40 - 0.22)/0.40 = 45\%$.

We can further improve the approximation, for example, by applying the technique proposed in [101]. Adding *white noise* to the flows of transitions of the TCPN model, we obtain the *continuous stochastic* CPN (TnCPN). In Fig.2.7, it can be clearly seen that the TnCPN model gives more accurate approximation to the original MPN: in the interval from 3.5 t.u. to 4.5 t.u., the average value of $\boldsymbol{m}[p_5]$ for the TnCPN is 0.34, the error is 15% (much better than the TCPN model with error of 45%).

One essential reason of the relatively inaccurate approximation of the deterministic model (Fig. 2.7) may be that the system is not truly very much populated: in $\boldsymbol{m}_0$, there is only one machine for each operation, and the size of buffers is also

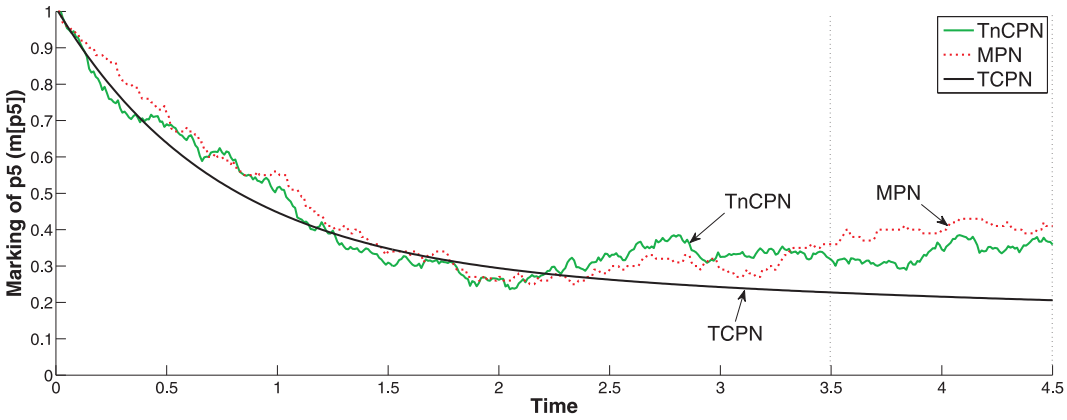


Figure 2.7: Marking trajectory of place p_5: with initial marking $\boldsymbol{m}_0$

limited to 1. In the case that the system is more populated, for example, instead of using $\boldsymbol{m}_0$, we simulate the system with initial marking equal to $50 \cdot \boldsymbol{m}_0$ (results shown in Fig.2.8), a very good approximation can already be obtained by using the deterministic TCPN model, even if no white noise is considered. More theoretical results about the approximation of using CPN can be found in [63].

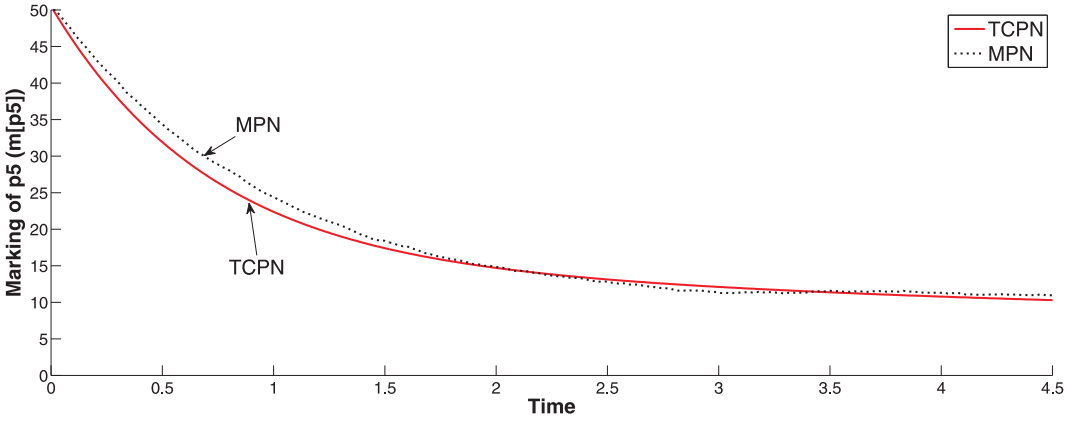


Figure 2.8: Marking trajectory of place p_5: with initial marking $50 \cdot \boldsymbol{m}_0$

Last but not least, let us consider the parametric optimization problem of computing the optimal initial marking that satisfies a linear constraint $\boldsymbol{K} \cdot \boldsymbol{m}_0 \leq \boldsymbol{d}$. Assume that because of constraints on the investment, we can have at most 5 machines in the system ($\boldsymbol{m}_0[M1\_Idle] + \boldsymbol{m}_0[M2\_Idle] + \boldsymbol{m}_0[M3\_Idle] \leq 5$), and the total size of buffers is constrained to no more than 10 ($\boldsymbol{m}_0[Max\_B\_1A] + \boldsymbol{m}_0[Max\_B\_1B] + \boldsymbol{m}_0[Max\_B\_3] \leq 10$). At the same time, the total amount of available pallets for the raw materials A and B are limited to 20 ($\boldsymbol{m}_0[Pallets\_A] + \boldsymbol{m}_0[Pallets\_B] \leq 20$). Under these constraints and with the other places initially set to be zero, we want to compute an optimal $\boldsymbol{m}_0$, such that the throughput of transition Out (t_{11}) in

the steady state is maximized. An optimal $\boldsymbol{m}_0$ obtained by solving LPP (2.13) is: $\boldsymbol{m}_0[Pallets\_A] = 3.4$, $\boldsymbol{m}_0[Pallets\_B] = 3.6$, $\boldsymbol{m}_0[Max\_B\_3] = 1.1$, $\boldsymbol{m}_0[Max\_B\_1A] = 2.0$, $\boldsymbol{m}_0[Max\_B\_1B] = 2.3$, $\boldsymbol{m}_0[M1\_Idle] = 2.1$, $\boldsymbol{m}_0[M2\_Idle] = 1.8$, $\boldsymbol{m}_0[M1\_Idle] = 1.1$. Using this initial marking, the maximal throughput of t_{11} is 0.2273.

Chapter 3

Control of Continuous Petri nets

This chapter recalls the main concepts and technical results related to the control of continuous Petri nets, which is the main topic of this thesis. We assume that all the transitions are controllable, and the system is controlled in the way that the flows of transitions can be slowed down. We focus on two control problems: *target marking control* and *optimal flow control*. Since in TCPNs under infinite server semantics the control inputs are non-negative and state-dependently bounded, classical results of the control of general continuous-state systems may not be directly applicable. Firstly, *controllability*, generally related to the capability of driving the system in a desired way, is discussed. Then, previous control methods are summarized. Some initial comparisons are also presented.

3.1 Introduction

3.1.1 Controlling the systems

In this section, we consider the systems under external control inputs (some dynamic control variables). The *flow* of transitions is interesting to be controlled. It is similar to the strategy which has been used for queuing networks, where servers activity and routing of customers are controlled (see, for example, [54]). We assume that the only admissible control action consists in *slowing down* the (maximal) firing flow of transitions (defined for the *uncontrolled* or *unforced* systems) [89]. This means that transitions modelling machines, for example, cannot work faster than their nominal speeds. Under this assumption, the *controlled* flow of a TCPN system is denoted as:

$$\boldsymbol{w}(\tau) = \boldsymbol{f}(\tau) - \boldsymbol{u}(\tau)$$

with $0 \leq \boldsymbol{u}(\tau) \leq \boldsymbol{f}(\tau)$ as the control inputs and $\boldsymbol{f}(\tau) = \boldsymbol{C}\boldsymbol{\Lambda}\boldsymbol{\Pi}(\boldsymbol{m}(\tau))\boldsymbol{m}(\tau)$ being the uncontrolled flow. Therefore, the overall behavior of the system is ruled by:

$$\dot{\boldsymbol{m}} = \boldsymbol{C} \cdot (\boldsymbol{f}(\tau) - \boldsymbol{u}(\tau))$$

A transition t_j is said to be *uncontrollable* if the only control input that can be applied is $\boldsymbol{u}(\tau)[t_j] = 0$. The transitions set T can be partitioned into disjoint sets of of *controllable* (T_c) and *uncontrollable* (T_{nc}) transitions, $T_c \cap T_{nc} = \emptyset$ and $T_c \cup T_{nc} = T$. In this thesis, we focus on the systems where all the transitions are controllable, i.e., $T_c = T$, $T_{nc} = \emptyset$.

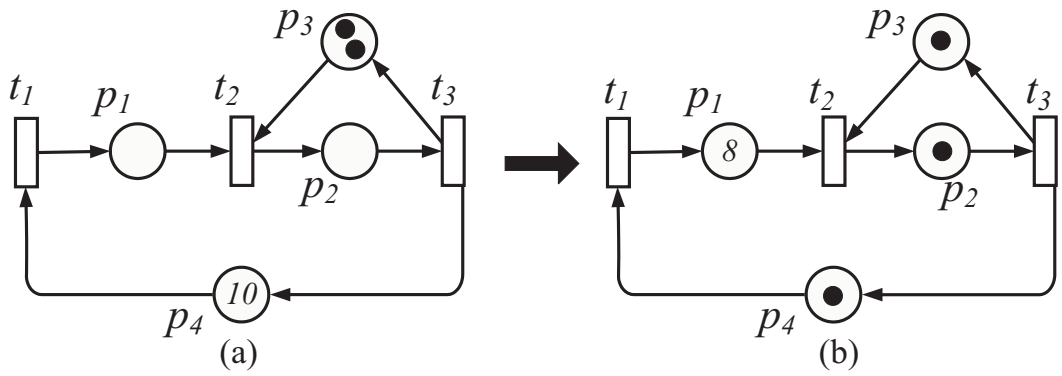
Many works can be found in the literature about the control of different classes of net systems. For instance, in the case of discrete PNs, *supervisory control* theory is studied (e.g. in [31, 38, 41]), in which the objective is to control the system behavior to satisfy certain (safety) specifications, for example, to avoid some *forbidden* states by disabling transitions in particular situations; or in the hybrid systems, e.g., aiming at optimizing a given objective function [9] or restricting the continuous reachable state space to a desired state space, which is expressed in terms of linear constraints only over the continuous variables [30]. Here, in TCPNs (under *infinite server semantics*), we focus on driving the system towards a desired steady state, or following certain state trajectories.

Among others, we will consider two related problems: (minimum-time) target marking control and (minimum-time) optimal flow control.

3.1.2 Target marking control problem

The target marking control problem concerns how to drive a PN system to a desired final state, denoted by $\boldsymbol{m}_f$, from a given initial state $\boldsymbol{m}_0$. In particular, we address the problem of reaching $\boldsymbol{m}_f$ in minimum-time. Then, we are able to maintain $\boldsymbol{m}_f$ (a steady-state) by using proper control inputs. For example, let us consider a very simple MG system, shown in Fig.3.1.

Figure 3.1: A simple MG system with firing rate vector $\boldsymbol{\lambda}$ =[1 1 1]^T: (a) initial state; and (b) desired final state



Assume that we want to drive the system to $\boldsymbol{m}_f$ =[8 1 1 1]^T (shown in Fig.3.1(b)), where the maximal flow of each transition can be obtained. For this particular example, if we simply let the system running "free", i.e., without applying any control ($\boldsymbol{u} = 0$), the system state will automatically evolve to $\boldsymbol{m}_f$ and then $\boldsymbol{m}_f$ is maintained, in around 6.1 time units. In order to reach $\boldsymbol{m}_f$, we can fire a sequence $t_1(9)t_2(1)$, then a simple (sequential) control law could be: first fire t_1 and block t_2, t_3 for 2.3 time units, until 9 tokens are put into p_1; then fire t_2 and block t_1, t_3, in 0.7 time units the final state is reached. Totally 3.0 time units are used, which is much faster than the one of without any control. However, this control strategy does not provide minimum-time state evolution to $\boldsymbol{m}_f$, because the firings of t_1 and t_2 are not necessary to be sequential. In Chapter 4, some centralized minimum-time control methods are proposed; in Chapter 5, 6 we will consider the problem in decentralized/distributed settings for large scale systems.

It is important to remark that this target marking control problem is similar to the *set-point* control problem, frequently addressed in general continuous-state systems. On the other hand, assuming that the continuous model approximates correctly the corresponding discrete one, it is analogous to reaching an average marking in the original discrete model. The control methods can be first developed in the continuous model, then applied to the original one. For example, a method for the control of open and closed manufacturing lines was proposed in [3]. Another related contribution can be found in [98], dealing with a stock-level control problem of an automotive assembling line system [27] originally modelled as a stochastic timed discrete PN. A framework of applying the control laws from the continuous model to the underlying discrete model was proposed, basically by applying additional delays to the controllable transitions.

3.1.3 Optimal flow control problem

Instead of driving the system to a given final state as in the target marking control problem, in the optimal flow control problem we focus on reaching an optimal flow in minimum-time. In particular, we are interested in a steady state where the maximal flow can be obtained. An important difference to the previous problem, also its main challenge, is that we may not be able to uniquely determine a final state, to which the system is driven. Consequently, we do not know which possible final state can be reached faster than the others—of course, it also depends on the control method being applied.

Let us consider the same system shown in Fig. 3.1, the maximal flow of transitions is equal to 1. It can be obtained in any marking that has at least one token in each place, for instance $\boldsymbol{m}_f{}' = [1\ 1\ 1\ 8]^T$. To reach $\boldsymbol{m}_f{}'$ we can fire a sequence $t_1(2)t_2(1)$. If we apply a similar sequential control strategy as we have applied for reaching $\boldsymbol{m}_f = [8\ 1\ 1\ 1]^T$ in the previous example, i.e., fire $t_1(2)$ and block other transitions; then fire $t_2(1)$, the system state reaches $\boldsymbol{m}_f{}'$ in only 0.91 time units. Therefore the maximal flow is achieved much faster (than in the case of reaching $\boldsymbol{m}_f$ with 3.0 time units). As we have mentioned, the time spent to reach a steady state with the maximal flow obviously depends on the applied control methods. For this MG, if we apply the ON/OFF controller presented in Chapter 4, then $\boldsymbol{m}_f{}'$ can be reached in only 0.81 time units. Moreover, we may still be able to further improve the time to reach the maximal flow (because $\boldsymbol{m}_f{}'$ may not be the "best" choice). We will address the (minimum-time) optimal flow control problem in Chapter 7.

3.2 Computing the initial and desired final states

3.2.1 About $\boldsymbol{m}_0$

In any practical system, for instance a production system, any transition should fire, therefore every place should be marked. In this thesis, we usually assume an initial state $\boldsymbol{m}_0 > 0$. With this assumption and if the net is consistent, the system is able to move in any direction of its reachability space [96], simplifying the computation of the control action. Even more, if at $\boldsymbol{m}_0$ some places are emptied and they are the support of a siphon, the net system is non-live and the final marking may not be reachable. For example, in the simple net shown in Fig. 2.2(a), provided with $\boldsymbol{m}_0 = [0\ 0\ 1]^T$ the system deadlocks. There exists no control law to reach a final state $\boldsymbol{m}_f = [0\ 1\ 0]^T$ even if $\boldsymbol{m}_f$ is a solution of the state equation with $\boldsymbol{\sigma} = [2\ 0\ 1]^T$ (the net system cannot move, in particular neither in the direction of $\boldsymbol{\sigma}$).

Let us assume that there exists no empty siphon at $\boldsymbol{m}_0 \geq 0$. This is a reasonable assumption in practice, otherwise, we can simply remove all the places in the empty siphons. Under this condition, if some places are empty, the system is able to reach a strictly positive marking easily. Nevertheless, since many solutions may exist, an open problem is to compute the most reasonable intermediate marking. This problem is not considered in this thesis.

3.2.2 About the desired/final state

As we have already mentioned, we consider two control problems: (1) *target marking control problem*, in which the desired marking is unique; and (2) *optimal flow control problem*, in which the desired markings belong to a convex region.

• For the target marking control problem: the final state $\boldsymbol{m}_f$ could be determined in a preliminarily *planning* stage, according to some optimality criteria [89], such as reducing the cost of resources or maximizing a benefit. A typical problem is to minimize the work-in-process (WIP) cost, trying to *maximize* the flow (throughput).

In a first step, the maximal weighted flow may be computed by solving the following LPP:

$$\begin{array}{ll} \psi = & \max\ \boldsymbol{g} \cdot \boldsymbol{w}_{ss} \\ \text{s.t.} & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_{ss} = \boldsymbol{0} \\ & \boldsymbol{w}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^\bullet t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_{ss}, \boldsymbol{\sigma}, \boldsymbol{m}_{ss} \geq \boldsymbol{0} \end{array} \tag{3.1}$$

where $\boldsymbol{v}[p_i,t]$ are *slack* variables; $\boldsymbol{m}_{ss}$ is a steady state marking, $\boldsymbol{w}_{ss}$ is the (controlled) flow in the steady-state, and $\boldsymbol{g}$ is a gain vector w.r.t. the flow.

It is similar to the parametric optimization problem that we have briefly recalled in Section 2.3.4, but remember that, due to the relaxation of *min* operator, the solution of LPP (2.13) and (2.14) for the unforced system gives, in general, an (not tight) upper bound of the optimal solution. However, in LPP (3.1) for a forced system assuming that all the transitions are controllable, by introducing the slack variable $\boldsymbol{v}[p_i,t_j]$, the original non-linear problem is transformed to a LPP. When p_i is the unique input place of t_j, variable $\boldsymbol{v}[p_i,t_j]$ can be viewed as the control input that reduces its flow. It is obvious that, if t_j is a synchronization, the minimal one, $\boldsymbol{u}[t_j] = \min_{p_i \in {}^\bullet t_j} \boldsymbol{v}[p_i,t_j]$, should be applied. More discussions about the optimal steady-state control problem of CPNs can be found in [65].

Since the maximal flow may be obtained in different steady states, then in a second step an optimal one ($\boldsymbol{m}_f = \boldsymbol{m}_{ss}$) with the minimal WIP cost, $\boldsymbol{l} \cdot \boldsymbol{m}_{ss}$ (where $\boldsymbol{l}$ is the work-in-process (WIP) cost vector), is computed by solving LPP:

$$\begin{array}{ll} \min & \boldsymbol{l} \cdot \boldsymbol{m}_{ss} \\ \text{s.t.} & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_{ss} = \boldsymbol{0} \\ & \boldsymbol{w}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^\bullet t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{g} \cdot \boldsymbol{w}_{ss} = \psi \\ & \boldsymbol{w}_{ss}, \boldsymbol{\sigma}, \boldsymbol{m}_{ss} \geq \boldsymbol{0} \end{array} \tag{3.2}$$

• For the *flow control problem*, the final state is not unique, belonging to a convex region in the reachability space of CPNs. All the states in this target convex region

can maximize certain profit functions, for example, they correspond to the maximal throughput (flow) of the system. Chapter 7 is devoted to this problem, in which instead of reaching a specific final marking, we are interested in reaching the maximal flow as fast as possible (without considering the WIP cost).

In TCPNs under infinite server semantics a marked place cannot be emptied in finite time (like the theoretical discharging of a capacitor in an electrical RC-circuit). Given a positive initial state $\boldsymbol{m}_0 > 0$, only a positive final state can be reached in finite time, thus the final state should also be an interior point in the reachability space, i.e., $\boldsymbol{m}_f > 0$.

3.3 Controllability

Controllability is an important property in every kind of dynamic systems. It is related to the capability of being driven in a certain desirable way and in this thesis, we consider the controllability in terms of the target marking control problem problem. More generally speaking, it is related to the classical controllability concept, according to which *a system is controllable* if for any two states $\boldsymbol{m}_1, \boldsymbol{m}_2$ of the state space it is possible to transfer the system from $\boldsymbol{m}_1$ to $\boldsymbol{m}_2$ in finite time (see, for instance, [21]).

A lot works can be found in the literature addressing the controllability of different classes of hybrid systems, for instance in [11, 33, 110]. However, in TCPNs, the control input are non-negative and state-dependently bounded, i.e., $\mathbf{0} \leq \boldsymbol{u} \leq \boldsymbol{\Lambda\Pi}(\boldsymbol{m})\boldsymbol{m}$, therefore the complexity of the analysis of controllability increases, and the classical controllability concept cannot be applied to TCPNs in general. Few contributions about the controllability of TCPNs only focused on very limited subclass, for example, JF nets [43]. Even by assuming that the control of the system is in a region such that the constraints are not active, systems are still not controllable due to the marking conservation laws imposed by P-flows [65]. More specifically, if $\boldsymbol{y}$ is a *P-flow* then any reachable marking $\boldsymbol{m}$ must fulfill $\boldsymbol{y}^T\boldsymbol{m} = \boldsymbol{y}^T\boldsymbol{m}_0$, defining thus a *state invariant*. Nevertheless, the study of controllability "over" this invariant is particularly interesting. This set is formally defined as $Class(\boldsymbol{m}_0) = \{\boldsymbol{m} \in \mathbb{R}^{|P|}_{\geq 0} | \boldsymbol{B}_y^T\boldsymbol{m} = \boldsymbol{B}_y^T\boldsymbol{m}_0\}$, where $\boldsymbol{B}_y$ is a basis of P-flows, i.e., $\boldsymbol{B}_y^T\boldsymbol{C} = \mathbf{0}$. For a general TCPN system, every reachable marking belongs to $Class(\boldsymbol{m}_0)$ (see Proposition 2.2.7).

Considering the constraints on the control input, an appropriate local controllability concept was proposed in [97]:

Definition 3.3.1. *The TCPN system* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ is controllable with bounded input *(BIC)* over $S \subseteq Class(\boldsymbol{m}_0)$ *if for any two markings* $\boldsymbol{m}_1, \boldsymbol{m}_2 \in S$ *there exists an input* $\boldsymbol{u}$ *transferring the system from* $\boldsymbol{m}_1$ *to* $\boldsymbol{m}_2$ *in finite or infinite time, and it is suitably bounded, i.e.,* $\mathbf{0} \leq \boldsymbol{u} \leq \boldsymbol{\Lambda\Pi}(\boldsymbol{m})\boldsymbol{m}$*, and* $\forall t_i \in T_{nc}$ $\boldsymbol{u}[t_i] = 0$ *along the marking trajectory.*

In the case that all the transitions in the system are controllable, the controllability of TCPNs only depends on the net structure, in particular, on the *consistency*.

Property 3.3.2. *[97] Let $\Sigma = \langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a TCPN system in which all the transitions are controllable. Σ is BIC over the interior of $Class(\boldsymbol{m}_0)$ iff $\mathcal{N}$ is* consistent. *Furthermore, the controllability is extended to the whole $Class(\boldsymbol{m}_0)$ iff (additionally to consistency) there exists no* empty siphon *at any marking in $Class(\boldsymbol{m}_0)$.*

Example 3.3.3. *Consider for instance the TCPN of Fig. 3.2(a) and the markings $\boldsymbol{m}_0 = [2\ 1\ 1]^T$, $\boldsymbol{m}_1 = [1\ 1\ 2]^T$ and $\boldsymbol{m}_2 = [1\ 2.5\ 0.5]^T$. Obviously, $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are both in $Class(\boldsymbol{m}_0)$. The system has only one P-semiflow (involving p_1, p_2 and p_3), the marking of two places is sufficient to represent the whole state. For this system $\exists \boldsymbol{\sigma} \geq \mathbf{0}$ such that $\boldsymbol{m}_1 = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$, but $\nexists \boldsymbol{\sigma} \geq \mathbf{0}$ such that $\boldsymbol{m}_2 = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. So $\boldsymbol{m}_1$ is reachable but $\boldsymbol{m}_2$ is not. It can be easily verified that the TCPN in Fig. 3.2(a) is not consistent, therefore according to Proposition 3.3.2 this TCPN is not controllable over $Class(\boldsymbol{m}_0)$. The shadowed area in Fig.3.2(a) corresponds to the set of reachable markings. It is the convex cone defined by vectors $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$, which represent the columns of $\boldsymbol{C}$ (here restricted to p_1 and p_3).*

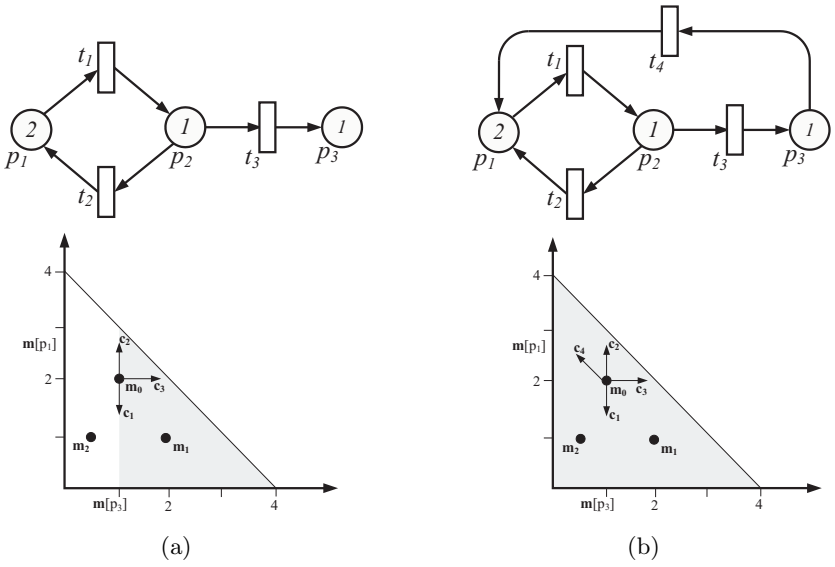


Figure 3.2: Two TCPN systems with identical P-flows and initial marking. The shadowed areas correspond to the sets of reachable markings. The net in (b) is consistent and there exists no empted siphon, therefore controllable over $Class(\boldsymbol{m}_0)$.

Now, consider the system of Fig. 3.2(b). In this case, $\boldsymbol{m}_2$ become reachable from $\boldsymbol{m}_0$. In fact, for any marking $\boldsymbol{m} \in Class(\boldsymbol{m}_0)$, the vector $(\boldsymbol{m} - \boldsymbol{m}_0)$ is in the convex cone defined by the vectors $\boldsymbol{c}_1$ to $\boldsymbol{c}_4$, which occurs due to the consistency of the net and implies that $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$. Moreover, since the only siphon in this net, composed of $\{p_1, p_2, p_3\}$, is always marked (at the same time it defines an initially marked conservative component), the system is BIC over $Class(\boldsymbol{m}_0)$.

If uncontrollable transitions exist, the analysis of controllability becomes more complex, and in general the systems are no longer controllable over $Class(\boldsymbol{m}_0)$, even for consistent nets (see [87] for some examples).

Since in the whole reachability space the system is usually uncontrollable when uncontrollable transitions exist, some contributions studied the controllability on the subsets of markings. For example, in [43], it is studied over the so called Controllability Space (CS), the set of all the *controllable markings*, that is characterized for Join-Free net. However, it is difficult to extend to general subclasses because its dependence on the markings. Contribution [97] focused on *equilibrium markings*. Marking $\boldsymbol{m}^q \in Class(\boldsymbol{m}_0)$ is an equilibrium one if $\exists \boldsymbol{u}^q$ $(0 \leq \boldsymbol{u}^q \leq \boldsymbol{\Lambda\Pi}(\boldsymbol{m}^q)\boldsymbol{m}^q)$ such that $\boldsymbol{C}(\boldsymbol{\Lambda\Pi}(\boldsymbol{m}^q)\boldsymbol{m}^q - \boldsymbol{u}^q) = \boldsymbol{0}$. They represent the *possible stationary operating points* of the system. The results are interesting, considering that controllers are frequently designed in order to drive the system towards a desired *stationary operating point*. Although here the technical results are not detailed, this approach is supported by the following proposition:

Property 3.3.4. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a TCPN system. Consider some equilibrium sets S_1, S_2,..., S_j related to different regions $\mathcal{R}_1$, $\mathcal{R}_2$,..., $\mathcal{R}_j$. If the system is BIC (in finite time) over each one and their union $\bigcup_{i=1}^{j} S_i$ is connected, the system is BIC over the union.*

Finally, let us mention that in the case of systems with uncontrollable transitions, the controllability may depend not only on the structure of the net, but also on the timing, more detailed explanations can be found in [87, 96].

3.4 Previous centralized control methods

Partially derived from [87], in this section we briefly summarize some control methods proposed in the literature for the target marking control problem of TCPNs. Some preliminary comparisons are presented. Notice that all the methods mentioned in this section are in the framework of centralized control, the decentralized/distributed control will be discussed in Chapter 5 and 6.

Most of the control methods that can be found in the literature assume that all the transitions are controllable:

Fuzzy control [36]

The authors proposed a control method for a particular variable speed CPNs, in which the firing speed v_j of a transition t_j is given by:

$$v_j = V_{jmax} \cdot \min\{1, m[p_{j1}], m[p_{j2}], ..., m[p_{jn}]\}$$

where $p_{j1}, p_{j2}, ..., p_{jn}$ are the input places of t_j and V_{jmax} is the maximal firing speed of t_j. It can be viewed as TCPNs under infinite server semantics with an implicit self-loop in each fluid transition. It is shown that the flow of a transition, can be represented as the output of two fuzzy rules under the Sugeno model. It was proved

that if the integral of the output of each fuzzy rule converges to a finite value then the resulting global fuzzy system (that represents the controlled flow) converges as well. Moreover, upper and lower bounds of this convergence were derived. Based on that, a *proportional fuzzy control* was proposed. Under a sufficient condition that the desired output (the marking of a place) is smaller than the initial upstream marking, it was proven that the convergence of the fuzzy global system can be obtained. However, this is not applicable to general cases.

Control for a piecewise-straight marking trajectory [44, 45, 5]

This approach was firstly explored in [44] for Join-Free nets, in which the tracking control problem of a mixed ramp-step reference signal is considered. Later, this method was extended to general PNs in [45]. There, a "high and low" gain proportional controller is synthesized, while a ramp-step reference trajectory, as a sort of *path-planning* problem at a higher level, is computed. To illustrate this kind of approach, let us detail a simple and more heuristic synthesis procedure introduced later in [5]. Consider the line l connecting $\boldsymbol{m}_0$ and $\boldsymbol{m}_d$, and the markings in the intersection of l with the region's borders, denoted as $\boldsymbol{m}_c^1$, $\boldsymbol{m}_c^2$, ..., $\boldsymbol{m}_c^n$. Define $\boldsymbol{m}_c^0 = \boldsymbol{m}_0$ and $\boldsymbol{m}_c^{n+1} = \boldsymbol{m}_f$. Then, $\forall k \in \{0, 1, ..., n\}$ compute τ_k by solving the linear programming problem (LPP):

$$\begin{array}{ll} min & \tau_k \\ s.t.: & \boldsymbol{m}_c^{k+1} = \boldsymbol{m}_c^k + \boldsymbol{C} \cdot \boldsymbol{x} \\ & \boldsymbol{0} \leq \boldsymbol{x}_j \leq \boldsymbol{\lambda}_j \boldsymbol{\Pi}_{ji}^k min\{\boldsymbol{m}_c^k[p_i], \boldsymbol{m}_c^{k+1}[p_i]\}\tau_k \\ & \forall j \in \{1, ..., |T|\} \text{ where i satisfies } \boldsymbol{\Pi}_{ji}^k \neq \boldsymbol{0} \end{array} \tag{3.3}$$

where $\boldsymbol{\Pi}^k$ is the configuration matrix corresponding to the region, to which $\boldsymbol{m}_c^k$ and $\boldsymbol{m}_c^{k+1}$ belong and $\boldsymbol{\Pi}_{ji}^k$ gives its element in the j^{th} row and i^{th} column.

The control law to be applied is thus $\boldsymbol{w} = \boldsymbol{x}/\tau_k$, when the system is between the markings $\boldsymbol{m}_c^k$ and $\boldsymbol{m}_c^{k+1}$. The time required for reaching the desired marking is given by $\tau_f = \sum_{k=0}^{n} \tau_k$. *Feasibility* and *convergence* to $\boldsymbol{m}_f$ were proved in [5].

In order to reach the final state faster, the trajectory is now not constrained to be straight linear, but piecewise-linear, i.e., only the states in the same region are constrained to be in a linear trajectory. The following *bilinear* programming problem (BPP) needs to be solved to find the *intermediate* states on the borders, reducing the accumulated time for reaching the final state.

$$\begin{array}{ll} \min & \tau_f = \sum_{k=0}^{n} \tau(k) \\ \boldsymbol{s.t} & \boldsymbol{m}^{k+1} = \boldsymbol{m}^k + \boldsymbol{C} \cdot \boldsymbol{x}^k, , k \in \{0, 1, ..., n\} \\ & (\Pi^k - \Pi^{k+1}) \cdot \boldsymbol{m}^k = 0, k \in \{1, 2, ..., n\} \\ & \boldsymbol{m}^k[p_i] \leq \boldsymbol{m}^{k+1}[p_i], \text{if } \boldsymbol{m}_0[p_i] \leq \boldsymbol{m}_f[p_i], p_i \in P, k \in \{0, 1, ..., n\} \\ & \boldsymbol{m}^k(p_i) \geq \boldsymbol{m}^{k+1}[p_i], \text{if } \boldsymbol{m}_0[p_i] \geq \boldsymbol{m}_f[p_i], p_i \in P, k \in \{0, 1, ..., n\} \\ & 0 \leq \boldsymbol{x}^k[t_j] \leq \lambda_j \cdot \Pi_{ji}^k \cdot \min\{\boldsymbol{m}^k[p_i], \boldsymbol{m}^{k+1}[p_i]\} \cdot \tau^k \\ & \quad \forall t_j \in T, \text{where } p_i \text{ satisfy } \Pi_{ji}^k \neq 0, k = \{0, 1, ...n\} \end{array} \tag{3.4}$$

Finally, by recursively solving similar BPPs as in (3.4), *intermediate states* are

added in the interior of each region, obtaining faster trajectories, until the accumulated time can not be significantly improved respect to a user specified threshold value.

Model predictive control (MPC) [64]

Model predictive control (MPC) has been widely applied in the industry for controlling complex dynamic systems [17, 70]. By solving a discrete-time optimal control problem over a given horizon, an optimal open-loop control input sequence is obtained and the first one is applied. Then in the next time step, a new optimal control problem is solved based on the current state and measurement, resulting in a *close-loop* control. In [64], the MPC scheme is applied to the control of TCPNs. The evolution of the timed continuous Petri net model, in *discrete-time*, is represented by the difference equation: $\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$, subject to the constraints $\mathbf{0} \leq \boldsymbol{w}_k \leq \boldsymbol{f}_k$ with $\boldsymbol{f}_k$ being the flow without control, which is equivalent to $\boldsymbol{G} \cdot [\boldsymbol{w}_k^T, \boldsymbol{m}_k^T]^T \leq \mathbf{0}$, for a particular matrix $\boldsymbol{G}$. The sampling period Θ must be chosen small enough in order to avoid spurious markings, in particular, for ensuring the positiveness of the markings. For that, the following condition is required to be fulfilled $\forall\, p \in P \;:\; \sum_{t_j \in p^\bullet} \lambda_j \Theta < 1$.

By using this representation of continuous PNs, in each time step the following optimization problem is solved:

$$
\begin{aligned}
\min \quad & J(\boldsymbol{m}_k, N) \\
s.t.: \quad & \boldsymbol{m}_{k+j+1} = \boldsymbol{m}_{k+j} + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_{k+j}, j = 0, ..., N-1 && \text{(3.5a)} \\
& \boldsymbol{G} \cdot \begin{bmatrix} \boldsymbol{w}_{k+j} \\ \boldsymbol{m}_{k+j} \end{bmatrix} \leq 0, j = 0, ..., N-1 && \text{(3.5b)} \\
& \boldsymbol{w}_{k+j} \geq 0, j = 0, ..., N-1 && \text{(3.5c)}
\end{aligned}
$$

where $J(\boldsymbol{m}_k, N)$ may be a quadratic objective function in the form of (3.6):

$$
\begin{aligned}
J(\boldsymbol{m}_k, N) = & (\boldsymbol{m}_{k+N} - \boldsymbol{m}_f)^T \cdot \boldsymbol{Z} \cdot (\boldsymbol{m}_{k+N} - \boldsymbol{m}_f) \\
& + \sum_{j=0}^{N-1} [(\boldsymbol{m}_{k+j} - \boldsymbol{m}_f)^T \cdot \boldsymbol{Q} \cdot (\boldsymbol{m}_{k+j} - \boldsymbol{m}_f) \\
& + (\boldsymbol{w}_{k+j} - \boldsymbol{w}_f)^T \cdot \boldsymbol{R} \cdot (\boldsymbol{w}_{k+j} - \boldsymbol{w}_{\boldsymbol{f}})]
\end{aligned}
\qquad \text{(3.6)}
$$

where $\boldsymbol{Z}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite matrices and N is a given time horizon, and $\boldsymbol{w}_f$ is a (desired) flow in the final state.

However, if the desired marking ($\boldsymbol{m}_f$) has zero components, the standard techniques used for ensuring converge in linear/hybrid systems (i.e., terminal constraints or terminal cost) cannot be applied in continuous nets [64]. Nevertheless, a particular control law is proposed to overcome this problem: the system state at time $k+N$ is constrained to the straight line from $\boldsymbol{m}_k$ to $\boldsymbol{m}_f$. Roughly, this is equivalent to add a terminal constraint in the form of:

$$\begin{cases} \boldsymbol{m}_{k+N} = \boldsymbol{m}_k + \alpha \cdot (\boldsymbol{m}_f - \boldsymbol{m}_k) \\ 0 \leq \alpha \leq 1 \end{cases} \tag{3.7}$$

where α is a new decision variable. The *asymptotic stability* of this method is proved in [64].

An alternative MPC approach for this problem is the so-called *explicit* solution [12], where the set of all states that are controllable is split into polytopes. In each polytope the control command is defined as a piecewise affine function of the state. The closed-loop *stability* is guaranteed with this approach. On the contrary, when either the order of the system or the length of the prediction horizon are not small, the *complexity* of the explicit controller becomes quickly prohibitive. Furthermore, the computation of the polytopes sometimes is infeasible.

Proportional control synthesis with LMI [51]
The proposed control scheme consists of a set of proportional (affine) control laws, one for each region. In detail, the controlled flow is represented, in discrete time, by $\boldsymbol{w}(k) = \boldsymbol{F}_r(\boldsymbol{m}(k) - \boldsymbol{m}_d) + \boldsymbol{R}$, where $\boldsymbol{R}$ is a vector and $\boldsymbol{F}_r$ is a gain matrix computed for each region (the subindex r denotes the $r-th$ region). In each region, the control and the marking are required to fulfill:

1. the input constraints: $\boldsymbol{0} \leq \boldsymbol{w}(k) \leq \boldsymbol{f}(k)$, where $\boldsymbol{f}(k)$ represents the flow without control,
2. the region membership: $\boldsymbol{m}(k) \in \mathcal{P}(\boldsymbol{G}_r, \boldsymbol{g}_r)$, where $\mathcal{P}(\boldsymbol{G}_r, \boldsymbol{g}_r) = \{\boldsymbol{m} | \boldsymbol{G}_r \boldsymbol{m} \leq \boldsymbol{g}_r\}$ is the inequality representation of the r-th region (a polyhedral),
3. the existence of a contractive invariant set (in order to prove closed-loop stability), which is stated as: $\boldsymbol{x}(k) \in \mathcal{P}(\boldsymbol{Q}, \boldsymbol{\mu}) \rightarrow \boldsymbol{x}(k+1) \in \mathcal{P}(\boldsymbol{Q}, \alpha\boldsymbol{\mu})$, where $\boldsymbol{x}(k) = (\boldsymbol{m}(k) - \boldsymbol{m}_d)$ is the current error, $\alpha < 1$ and $\mathcal{P}(\boldsymbol{Q}, \alpha\boldsymbol{\mu}) = \{\boldsymbol{x} | \boldsymbol{Q}\boldsymbol{x} \leq \alpha\boldsymbol{\mu}\}$ is the contractive set (so, the absolute error is monotonicdecreasing).

The methodology consists in expressing the previous conditions as sets of linear matrix inequalities (LMI), one set for each region. The solution of a LMI can be achieved in polynomial time. Furthermore, *convergence* to the desired marking $\boldsymbol{m}_d$ is guaranteed. The main drawback of this approach is that a LMI must be solved for each region, but the number of these increases exponentially w.r.t. the number of synchronizations (joins).

Affine control [102]
The synthesis of controllers for TCPNs can be geometrically expressed in terms of polytopes. An affine system in polytopes χ is defined as:

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{a}$$

with restriction $x \in \chi$ and $\boldsymbol{u} \in U$, where U is a polytope of admissible inputs. An admissible *affine control law* is a affine function $\boldsymbol{u} : \chi \rightarrow U$, characterized by $\boldsymbol{u}(x) = \boldsymbol{F}x + \boldsymbol{g}$. The affine control is used in the synthesis of piecewise hybrid

system in [34, 32], by decomposing the polytopes into simplices and synthesizing a proper affine control law for each of them. In [102] this method is extended to the control of TCPNs, in which global affine control laws for the complete polytopes are synthesized. The vertices of a polytope of dimension $k-1$ are enumerated, and it is assume that the first k vertices define a simplex of dimension $k-1$. Then the evaluation of the control law at the vertices and conditions for the unique equilibrium point (in close loop) are derived. It is proved that given a consistent TCPN with initial state $\boldsymbol{m}_0 > 0$, using this affine control technology, the system can always be driven to a desirable final state $\boldsymbol{m}_f > 0$. The main drawback of this method is that the number of vertices increases exponentially respect to the number of dimensions that is determined by the number of places, therefore its computational complexity may be intractable (although it can be partially done off-line).

In this work, we assume all the transitions are controllable. In the case that uncontrollable transitions exist, the control problem becomes much more complex. Few works can be found in the literatures considering partially controllable systems. For instance, in [58], a Gradient-base control based method was proposed; anther method that considered uncontrollable transitions is Pole assignment control proposed in [99], where the initial and desired markings are equilibrium states.

3.4.1 Initial comparisons

The availability of many control methods for this target marking control problem of TCPNs makes difficult the selection of the most appropriate technique for a given system and purpose. In order to make an appropriate choice, several properties may be taken into account, e.g., feasibility, closed-loop stability, robustness, computational complexity (for the synthesis and during the applications), etc.

Table 3.1: Qualitative characteristics of several control methods (assuming $\boldsymbol{m}_0 > 0$, $\boldsymbol{m}_f > 0$ and all transitions are controllable). The following abbreviations are used: min. (minimize), suff. (sufficient conditions), comput. (computational), quad. (quadratic) and poly. (polynomial).

Methods	**Comput. issues**	**Optimizing index**	**Stability**
Fuzzy control	two fuzzy rules per transition	None	under suff.
Piecewise-straight trajectory	LPPs or BPPs on $\|T\|$	Heuristic Min. time	Yes
MPC	QPPs on $\|T\|, N$	Min. quad. (or linear) functions	under suff.
LMI	A LMI for each configuration	None	under suff.
Affine Control	Expon. on $\|P\|$	None	Yes

Table 3.1 (that partially derived from Table 4 in [87]) demonstrates some qualitative properties of different control methods (under infinite server semantics) that have been described, assuming all the transitions are controllable. All those methods are applicable to any PN structure. The fuzzy control guarantees the convergence based on some sufficient conditions that may be too "restrictive" in general cases. The MPC based approaches ensure convergence and minimize a quadratic or linear objective function, obtaining a desired state trajectory. Nevertheless, when the number of transitions grows, or a large time horizon N is considered, its complexity for solving the problem with a huge number of variables may become intractable. In such cases, the piecewise-straight trajectory method could be more appropriate. However, when the process for obtaining heuristic minimum-time evolution is considered, the computational complexity is also very high. For instance, in the method proposed in [5], a non-linear BPP problem needs to be solved once an intermediate state is introduced to reduce the time. Affine controller also guarantees the convergence to the final state, but no optimizing index is considered; on the other hand, its complexity may increase rapidly in a larger system with many places.

Now let us consider some a few examples using different methods for the target marking control problem. The simulations are performed by using Matlab 8.0 on a PC with Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz, 3.24GB of RAM.

This first net system we consider is shown in Fig. 3.3. Let us assume that the firing rate of every transition is equal to 1; the sampling period is $\Theta = 0.01$. We will consider two different initial states $\boldsymbol{m}_{01} = [3\ 3\ 1\ 3]^T$; $\boldsymbol{m}_{02} = [2.1\ 2.1\ 0.1\ 2.1]^T$ and also two different final marking: $\boldsymbol{m}_{f_1} = [1\ 4.5\ 1.5\ 3]^T$ and $\boldsymbol{m}_{f_2} = [0.1\ 3.6\ 0.6\ 2.1]^T$, respectively. It can be checked that $\boldsymbol{m}_{f_i}$ can be reached from $\boldsymbol{m}_{0i}$, $i = 1, 2$ with the same firing count vectors, but one element of $\boldsymbol{m}_{02}$ is very small. We can observe later that the performance of some control methods is quite dependent on the initial marking.

For this control example we have applied the approaching minimum-time controller (appro. min-time) [5], affine controller [102] and the MPC controller [64]. Different parameters of the MPC controller are used. The simulation results are shown in Table 3.2 (considering $\boldsymbol{m}_{01}/\boldsymbol{m}_{f_1}$) and Table 3.3 (considering $\boldsymbol{m}_{02}/\boldsymbol{m}_{f_2}$).

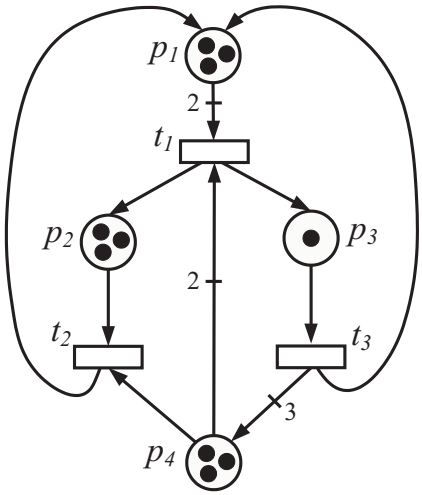


Figure 3.3: A simple TCPN system

Table 3.2: Simulation results of reaching $\boldsymbol{m}_{f_1}$ from $\boldsymbol{m}_{01}$, in the net system of Fig. 3.3 (for the MPC control, the weight matrix $\boldsymbol{Q} = q \cdot \mathbf{1}^{|P|}$, $\boldsymbol{R} = r \cdot \mathbf{1}^{|T|}$)

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	176	402	
affine control*	1,639	102	
MPC control	173	630	$N = 1$, q = 1000, r = 0.2
MPC control	185	684	$N = 1$, q = 1000, r = 2
MPC control	249	885	$N = 1$, q = 1000, r = 20
MPC control	173	1,126	$N = 3$, q = 1000, r = 0.2
MPC control	182	1,157	$N = 3$, q = 1000, r = 2
MPC control	236	1,385	$N = 3$, q = 1000, r = 20
MPC control	173	1,929	$N = 5$, q = 1000, r = 0.2
MPC control	181	2,002	$N = 5$, q = 1000, r = 2
MPC control	228	2,260	$N = 5$, q = 1000, r = 20
MPC control	173	5,099	$N = 10$, q = 1000, r = 0.2
MPC control	180	5,318	$N = 10$, q = 1000, r = 2
MPC control	219	5,816	$N = 10$, q = 1000, r = 20

*The affine control is not designed for minimum-time control, and there may exist many optimizing parameters, but it is not clear how to choose one to minimize the time. Here, no optimizing parameter is used.

We can observe in this example that in the case of reaching $\boldsymbol{m}_{f_1}$ from $\boldsymbol{m}_{01}$, the approaching minimum-time controller gives a number of time steps slightly larger than that of the MPC controller; but its computational cost is less. However, in the case of reaching $\boldsymbol{m}_{f_2}$ from $\boldsymbol{m}_{02}$ (Table 3.3), the approaching minimum-time controller does not work very well. Because in this approach, between each pair

Table 3.3: Simulation results of reaching $\boldsymbol{m}_{f_2}$ from $\boldsymbol{m}_{02}$, in the net system of Fig. 3.3 (for the MPC control, the weight matrix $\boldsymbol{Q} = q \cdot \mathbf{1}^{|P|}$, $\boldsymbol{R} = r \cdot \mathbf{1}^{|T|}$)

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	1,076	490	
affine control*	6,454	356	
MPC control	490	1,833	$N = 1$, q = 1000, r = 0.2
MPC control	486	1,811	$N = 1$, q = 1000, r = 2
MPC control	517	1,880	$N = 1$, q = 1000, r = 20
MPC control	493	3,203	$N = 3$, q = 1000, r = 0.2
MPC control	492	3,193	$N = 3$, q = 1000, r = 2
MPC control	514	3,214	$N = 3$, q = 1000, r = 20
MPC control	493	5,565	$N = 5$, q = 1000, r = 0.2
MPC control	494	5,572	$N = 5$, q = 1000, r = 2
MPC control	512	5,571	$N = 5$, q = 1000, r = 20
MPC control	492	14,916	$N = 10$, q = 1000, r = 0.2
MPC control	493	14,887	$N = 10$, q = 1000, r = 2
MPC control	509	14,962	$N = 10$, q = 1000, r = 20

*The affine control is not designed for minimum-time control, and there may exist many optimizing parameters, but it is not clear how to choose one to minimize the time. Here, no optimizing parameter is used.

of adjacent states of the trajectory the firing speed is constant and determined by the one with smaller flow; therefore, if one of the states has very small flow (in this case, the initial one), the time spent for reaching $\boldsymbol{m}_f$ could be large. The affine controller costs more time steps to reach the final state, because it is not designed for the minimum-time control. For the MPC controller, both Tables show that a small number of time steps could be obtained by using a large weight for matrix $\boldsymbol{Q}$ and a small weight for $\boldsymbol{R}$. We should also notice that the MPC controller is not designed for minimum-time evolution either, and using a larger time horizon N does not guarantee a smaller time to reach the final state.

Now let us consider a larger net system in Fig. 2.6 that we have discussed in Section 2.4. We assume a positive initial state (required by the control methods) that each of the emptied place in Fig. 2.6 has marking equal to 0.1, and for the other places we keep their markings as in Fig. 2.6. Let us assume that we want to reach a final state that obtains the maximal flow $\psi = 0.13$ (computed by solving a LPP similar to (3.1)), $\boldsymbol{m}_f = [18.78\ 0.39\ 0.13\ 0.52\ 0.16\ 0.29\ 13.65\ 0.39\ 0.13\ 0.65\ 0.13\ 0.13\ 0.58\ 0.52\ 0.97\ 0.13\ 0.26\ 0.13]^T$. The simulation results are shown in Table 3.4.

As we have already mentioned, in affine control the number of vertices increases exponentially with respect to the number of dimensions that is determined by the number of places; therefore its computational complexity may easily be intractable. In the PC that we do the simulation, the computational cost of the affine controller is intractable for the net system of Fig. 2.6 . In this example, the smallest number

Table 3.4: Simulation results in the net system of Fig. 2.6 (for the MPC control, the weight matrix $\boldsymbol{Q} = q \cdot \mathbf{1}^{|P|}$, $\boldsymbol{R} = r \cdot \mathbf{1}^{|T|}$)

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	875	6,584	
affine control	NA	NA	
MPC control	678	6,339	$N = 1$, q = 1000, r = 0.2
MPC control	724	6,396	$N = 1$, q = 1000, r = 2
MPC control	984	7,697	$N = 1$, q = 1000, r = 20
MPC control	682	32,391	$N = 3$, q = 1000, r = 0.2
MPC control	703	29,641	$N = 3$, q = 1000, r = 2
MPC control	920	30,958	$N = 3$, q = 1000, r = 20
MPC control	683	81,614	$N = 5$, q = 1000, r = 0.2
MPC control	695	74,965	$N = 5$, q = 1000, r = 2
MPC control	879	80,076	$N = 5$, q = 1000, r = 20
MPC control	677	481,274	$N = 10$, q = 1000, r = 0.2
MPC control	686	427,351	$N = 10$, q = 1000, r = 2
MPC control	818	415,506	$N = 10$, q = 1000, r = 20

of time steps is obtained by using the MPC controller. However it has very high computational costs when N increases (when $N = 10$, its consumed CPU time is almost 100 times as large as the one of the approaching minimum-time controller). We can also observe that the number of time steps does not improve quickly by using a larger N, so a smaller N may be a reasonable choice.

3.5 Conclusions

In this chapter, we review the basic concepts, theories and methodologies about the control of TCPNs, mainly under infinite server server semantics. In this thesis, we consider two control problems:

- *target marking control* problem—driving the system to a given desired final state from an initial one;

- *optimal flow control* problem—driving the system to an optimal flow (obtained in a convex region).

Most of the work in the literature related to the control of TCPNs are devoted to the first one, which is similar to the typical set-point control problem in a general continuous-state system. The controllability—the capability of being driven in a certain desired way, in particular, moving form one state to another, is reviewed. We also briefly recall the existing (centralized) methods for the target marking control problem, and an initial comparison of some qualitative properties of several different control methods are given in Table 3.1.

Regarding to the previous works, we may conclude: 1) the computational complexity of many of the described control methods may increase very fast, even exponentially, with respect to the size of the system (for example the affine control proposed in [102]); 2) for the target marking control problem, one important goal is to reach the desired final state as fast as possible, but this minimum-time problem has not been addressed in most of the existing methods; 3) most of the contributions focus on the centralized control, it may be interesting to consider the problem in decentralized environments. Although few work can be found in the literature, (eg., in [4], assuming nets to be mono-T-semiflow), it is still very limited; 4) to the best knowledge of the author, the (minimum-time) optimal flow control problem of TCPNs has not be studied. In the following chapters, those problems will be addressed.

Chapter 4

Centralized Control: ON/OFF Based Methods

This chapter focuses on centralized methods for the target marking control problem, addressing minimum-time evolution to the desired final state. In particular, several ON/OFF based controllers (or Bang-Bang based controllers that frequently arise in optimal control) are presented. First, we propose a (standard) ON/OFF controller for Choice-Free (CF) net systems and prove that it is a minimum-time controller driving the system to the final state. However, an illustrative example shows that the standard ON/OFF control strategy is not "fair" for solving the conflicts and may "impose deadlocked" situations even to a live and bounded system. In order to overcome this problem, we introduce some heuristic strategies for solving the conflicts, obtaining three extended controllers for general nets: ON/OFF+, B-ON/OFF and MPC-ON/OFF. We also give an algorithm to compute the minimum-time control law, but it may easily become intractable for a large system because of its high computational complexity. Finally, we demonstrate the proposed control methods with examples. More case studies are presented in Chapter 8.

4.1 Motivation: minimum-time state evolution

Optimal control [77, 6, 14] deals with the problem of finding a control law for a given system such that a certain optimality criterion, formulated as a cost function of state and control variables, is achieved. Among other objectives, minimum-time control has been widely studied (see, for example, [52, 86, 15]).

In this Chapter we focus on the minimum-time target marking control problem of TCPNs under infinite server semantics, which can be simply represented as follows [89]:

$$\begin{array}{ll} \min & \tau \\ s.t. & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \int_0^\tau \boldsymbol{w}(\delta)\mathrm{d}\delta \\ & \boldsymbol{w}(\delta)[t_j] = \boldsymbol{\lambda}[t_j] \cdot \min_{p_i \in {}^\bullet t_j} \{\frac{\boldsymbol{m}(\delta)[p_i]}{\boldsymbol{Pre}[p_i,t_j]}\} - \boldsymbol{u}(\delta)[t_j], \forall t_j \\ & \boldsymbol{m}(\delta), \boldsymbol{w}(\delta), \boldsymbol{u}(\delta) \geq 0 \end{array} \tag{4.1}$$

where $\boldsymbol{u}(\delta)$, $\boldsymbol{w}(\delta)$ are the control input and controlled flow at time δ.

In general, problem (4.1) is difficult to solve because of the simultaneous existence of minimum operators and state (marking) dependent constraints for the control variables. Except for the heuristic minimum-time controller proposed in [5], among the control methods for TCPNs we have mentioned in Chapter 3, they only address the convergence to the final state. Moreover, the MPC controller can be used to optimize the state trajectory, but minimum-time evolution is not guaranteed and this is very difficult to approach in the general MPC framework. Actually, the time spent for reaching a desired final state by applying different control methods may vary significantly, for example as shown in the control examples of Section 3.4.1.

An ON/OFF (or Bang-Bang) controller, is a feedback controller that switches actions from one extreme to the other at certain times (switching points). Regarding the optimal control, ON/OFF strategies frequently arise in minimum-time problems. A very simple example is to drive a car to a desired position (in a straight line) in shortest time—the solution is to apply the maximum acceleration until a unique switching point, then apply the maximum breaking and stop the car exactly at the desired position. Other common applications of the ON/OFF controller include residential thermostats, process of boiling water, etc.

In the following sections, we propose several ON/OFF based controllers. We prove that for some subclasses like CF nets, minimum-time state evolution is ensured; for general nets systems, we present heuristic algorithms. For the standard ON/OFF controller of CF nets, a positive initial state (i.e., $\boldsymbol{m}_0 > 0$) is not mandatory; while for the extended controllers of general nets, we assume $\boldsymbol{m}_0 > 0$. We always assume a positive final state (i.e., $\boldsymbol{m}_f > 0$), because in TCPNs under infinite server semantics it takes infinite time to empty a marked place. The main advantage of our methods is that the computational complexity is very low and a reasonable number of time steps for reaching the final state can be obtained.

4.2 Minimum-time controller for Choice-Free nets

In this section we propose an ON/OFF controller for CF nets: every transition fires as fast as possible until a given upper bound, the *minimal firing count vector*, is reached. We prove that this very simple ON/OFF control strategy drives the system to the desired final state in minimum-time. A manufacturing system is used to illustrate the proposed method.

4.2.1 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 by firing σ, it is also reached by 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 for CF nets it is unique under some general assumptions .

Definition 4.2.1. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a CPN system and $\boldsymbol{m}_f$ be a reachable marking through a sequence σ, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. A firing count vector $\boldsymbol{\sigma}$ is said to be* minimal *if for any T-semiflow $\boldsymbol{x}$, $||\boldsymbol{x}|| \not\subseteq ||\boldsymbol{\sigma}||$, where $||\cdot||$ stands for the support of a vector. We can simply compute a $\boldsymbol{\sigma}$ by solving the following LPP:*

$$\begin{array}{ll} min & \mathbf{1}^T \cdot \boldsymbol{\sigma} \\ s.t. & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{\sigma} \geq 0 \end{array} \tag{4.2}$$

Example 4.2.2. *The minimal firing count vector may not be unique for a non-CF net. For example, let us consider the non-CF net in Fig. 4.1. Assume that $\boldsymbol{m}_0 = [4\ 0\ 0\ 0\ 0]^T$ and $\boldsymbol{m}_f = [2\ 0\ 0\ 0\ 1]^T$. To drive the system to its final state, there exist two minimal firing count vectors $\boldsymbol{\sigma}_1 = [1\ 1\ 0\ 1\ 0\ 0]^T$ and $\boldsymbol{\sigma_2} = [0\ 0\ 1\ 0\ 1\ 0]^T$. The final state can also be reached by firing $\boldsymbol{\sigma}_3 = [1\ 1\ 1\ 1\ 1\ 1]^T$, but $\boldsymbol{\sigma}_3$ is not a minimal firing count vector, because it contains a T-semiflow $[0\ 0\ 1\ 0\ 1\ 1]^T$.*

Proposition 4.2.3. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a CF net system and $\boldsymbol{m}_f$ be a reachable marking. If one of the following assumptions holds, there exits a unique minimal firing count vector $\boldsymbol{\sigma}$.*

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

(A2) The net is strongly connected and consistent.

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

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

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

If (A2) holds, there is only one minimal T-semiflow [93], denoted by $\boldsymbol{x} > 0$. $\boldsymbol{\sigma}_{12}$ may have negative elements, but we can always find an $\alpha \geq 0$, such that $\boldsymbol{\sigma}_{12} + \alpha \cdot \boldsymbol{x} \geq$

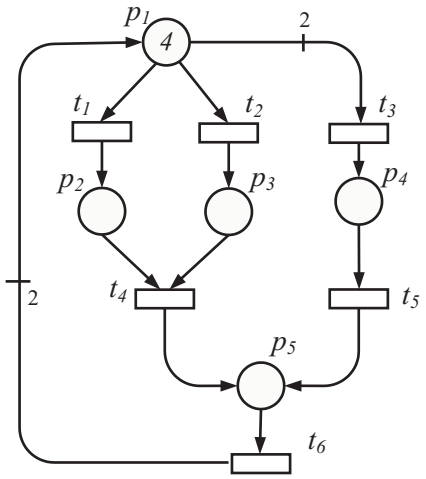


Figure 4.1: A non-CF Petri net system

0. Since $\boldsymbol{C} \cdot (\boldsymbol{\sigma}_{12} + \alpha \cdot \boldsymbol{x}) = 0$ and $\boldsymbol{\sigma}_{12} + \alpha \cdot \boldsymbol{x} \geq 0$, it is a T-semiflow. Therefore, there exists $\beta > 0$ such that $\boldsymbol{\sigma}_{12} + \alpha \cdot \boldsymbol{x} = \beta \cdot \boldsymbol{x}$, implying $\boldsymbol{\sigma}_{12} = (\beta - \alpha) \cdot \boldsymbol{x}$. If $\beta - \alpha = 0$ then $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2$ which is impossible by assumption. If $\beta - \alpha > 0$ then $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 + (\beta - \alpha) \cdot \boldsymbol{x} > (\beta - \alpha) \cdot \boldsymbol{x}$. Therefore, $\boldsymbol{\sigma}_1$ is not a minimal firing count vector. Similarly, if $\beta - \alpha < 0$ then $\boldsymbol{\sigma}_2$ is not a minimal firing count vector. ■

In the sequel, we assume strongly connected and consistent CF nets. Thus, any controller driving the system to $\boldsymbol{m}_f$ must follow *the* minimal firing count vector plus eventually a T-semiflow. We will prove that by using the minimal firing count and applying an *ON/OFF controller*, $\boldsymbol{m}_f$ can be reached in minimum-time.

4.2.2 ON/OFF controller: discrete-time case

As already said in Section 3.4, by sampling the continuous-time CPN system with a *sampling period* Θ, we obtain the discrete-time TCPN ([64]) given by:

$$\begin{aligned} \boldsymbol{m}_{k+1} &= \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k \\ 0 &\leq \boldsymbol{w}_k \leq \boldsymbol{f}_k \end{aligned} \tag{4.3}$$

Here $\boldsymbol{m}_k$ and $\boldsymbol{w}_k = \boldsymbol{f}_k - \boldsymbol{u}_k$ are the marking and controlled flow at sampling instant k, i.e., at $\tau = k \cdot \Theta$, while $\boldsymbol{f}_k$ and $\boldsymbol{u}_k$ are the uncontrolled flow and control input.

It is proved in [64] that if the sampling period satisfies (4.4), the interior reachability spaces of discrete-time and continuous-time CPN systems are the same.

$$\forall p \in P : \sum_{t_j \in p^\bullet} \lambda_j \cdot \Theta < 1 \tag{4.4}$$

In the sequel, we assume that the sampling period Θ is small enough to satisfy (4.4). We first develop the ON/OFF controller based on the discrete-time model, then it is naturally extended to continuous-time settings.

In a CF net system, if two transitions t_1 and t_2 are enabled at the same time, the order of firing is not important (i.e., both sequence t_1t_2 and t_2t_1 are fireable). Based on this observation, if there exists a transition that has not fired with the maximal amount at one moment, certain amount of its firings may be moved ahead in order to reach this maximal quantity.

Example 4.2.4. *Let us consider the trivial CF net system in Fig. 4.2 and assume* $\boldsymbol{m}_f = [0.2\ 0.5\ 0.3]^T$*, the minimal firing count vector for reaching the final state is* $\boldsymbol{\sigma} = [0.8\ 0.3\ 0]^T$*. Following this vector, one firing sequence may be* $\sigma_1 = t_1(0.5)t_2(0.3)t_1(0.3)$*. It can be observed that* t_1 *is 1-enabled under* $\boldsymbol{m}_0$*, and the required amount that* t_1 *should fire is* 0.8*. Therefore, we can fire* t_1 *more than* 0.5 *in the beginning. In particular, the final marking is also reached by the firing sequence* $\sigma_2 = t_1(0.8)t_2(0.3)$*, for example.*

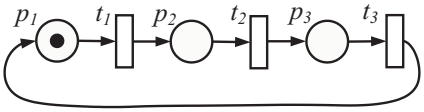


Figure 4.2: A trivial CF net system with $\boldsymbol{m}_0 = [1\ 0\ 0]$.

The strategy of the ON/OFF controller is quite simple: every transition fires as fast as possible at any moment until the required firing count $\boldsymbol{\sigma}[t_j]$ ($\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C}\cdot\boldsymbol{\sigma}$) is reached. The control input of t_j at k^{th} sampling period is:

$$\boldsymbol{u}_k[t_j] = \begin{cases} 0 & \text{if} \quad \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j] + \Theta \cdot \boldsymbol{f}_k[t_j] \le \boldsymbol{\sigma}[t_j] \quad \text{(a)} \\ \boldsymbol{f}_k[t_j] & \text{if} \quad \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j] = \boldsymbol{\sigma}[t_j] \quad \text{(b)} \\ \boldsymbol{f}_k[t_j] - \dfrac{\boldsymbol{\sigma}[t_j] - \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j]}{\Theta} & \\ & \text{if} \quad \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j] < \boldsymbol{\sigma}[t_j] \text{ and} \quad \text{(c)} \\ & \quad \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j] + \Theta \cdot \boldsymbol{f}[t_j] > \boldsymbol{\sigma}[t_j] \end{cases} \tag{4.5}$$

where at $k = 0$, $\Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i[t_j] = 0$. Remember $\boldsymbol{w}_k = \boldsymbol{f}_k - \boldsymbol{u}_k$, (a) says that before reaching the required total firing count $\boldsymbol{\sigma}[t_j]$, we simply let transition t_j to fire *free* (ON), i.e. $\boldsymbol{u}_k[t_j] = 0$; (b) means once $\boldsymbol{\sigma}[t_j]$ is reached, the transition is completely stopped (OFF), i.e. $\boldsymbol{u}_k[t_j] = \boldsymbol{f}_k[t_j]$; (c) represents the last firing of t_j. Algorithm 1 synthesizes the ON/OFF controller, in which $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ... is the sequence of control inputs at the time instants.

Lemma 4.2.5. *Let* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *be a discrete-time continuous CF net system and* $\boldsymbol{m}_f > 0$ *be a reachable final state. Among all the controllers that drive the system to*

Algorithm 1 ON/OFF controller

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{\sigma}$, Θ
Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

1: $k = 0$
2: **while** $\Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i \leq \boldsymbol{\sigma}$ **do**
3: *Solve the following LPP :*

$$\begin{aligned} max \quad & \mathbf{1}^T \cdot \boldsymbol{w}_k \\ \textbf{s.t.} \quad & \boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k \\ & \mathbf{0} \leq \Theta \cdot \boldsymbol{w}_k \leq \boldsymbol{\sigma} - \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i \\ & \boldsymbol{w}_k[t_j] \leq \lambda_j \cdot enab(t_j, \boldsymbol{m}_k), \forall t_j \in T \end{aligned} \tag{4.6}$$

4: *Apply* $\boldsymbol{w}_k : \boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$
5: $k := k + 1$
6: **end while**
7: **return** $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

$\boldsymbol{m}_f$ by firing $\boldsymbol{\sigma}$, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$, the ON/OFF controller costs the minimum-time.

Proof: Assume an arbitrary non ON/OFF controller $\boldsymbol{G}$. Hence, at a sampling period k there exists a transition t_j that is not *sufficiently fired*, i.e., not fired as much as possible. In other words, t_j has to fire later in a sampling period l, $l > k$. Let us assume, without loss of generality, that t_j does not fire between the k^{th} and the l^{th} sampling periods. It is always possible to "move" some amounts of its firings from the l^{th} sampling period to the k^{th} one until t_j becomes sufficiently fired in k. According to the persistency property of CF nets, this move is not reducing the enabling degree of the other transitions. Iterating the procedure, all transitions can be sufficiently fired in all sampling periods and the obtained controller is an ON/OFF one. Obviously, the number of discrete-time periods required to reach the final marking after moving firings from a sampling period l to another one k with $k \leq l$ is at least the same. Hence the number of sampling steps of the ON/OFF controller is not more than the one of controller $\boldsymbol{G}$, i.e., the ON/OFF controller costs the minimum time. ■

Lemma 4.2.5 only holds for CF nets. For a net system that is not CF, the ON/OFF controller may be not a minimum-time controller for a given $\boldsymbol{\sigma}$, and in the worst case, the final state may not be reached, i.e., the stability is not guaranteed (see Ex. 4.3.1 for an example, in which the (reachable) final state cannot be reached by applying the ON/OFF controller using the given $\boldsymbol{\sigma}$.)

Lemma 4.2.6. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a discrete-time continuous CF net system, $\boldsymbol{\sigma}$*

and $\boldsymbol{\sigma}'$ be firing count vectors, such that $\boldsymbol{\sigma} \leq \boldsymbol{\sigma}'$ and they both drive the system to $\boldsymbol{m}_f > 0$, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$ and $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}'$. By using the ON/OFF controller, firing $\boldsymbol{\sigma}'$ costs at least the time of firing $\boldsymbol{\sigma}$.

Proof: If firing $\boldsymbol{\sigma}'$ costs less time than firing $\boldsymbol{\sigma}$, there must exist at least a transition t_j, such that firing $\boldsymbol{\sigma}'[t_j]$ costs less time than firing $\boldsymbol{\sigma}[t_j]$. We will prove that it is not possible.

Since by firing $\boldsymbol{\sigma}'$ and $\boldsymbol{\sigma}$ the same final state is reached, $\boldsymbol{\sigma}'[t_j] \geq \boldsymbol{\sigma}[t_j]$ implies that in the case of firing $\boldsymbol{\sigma}'$, more tokens should be put into each of its input place $p_i \in {}^\bullet t_j$ (with the quantity of $\boldsymbol{Pre}[p_i, t_j] \cdot (\boldsymbol{\sigma}'[t_j] - \boldsymbol{\sigma}[t_j])$), and later they are all moved out (by firing t_j). Remember that t_j is the unique output transition of p_i (the net is CF), so the time spent for moving out this quantity of tokens depends only on t_j; and since this quantity of tokens have to be moved out, they do not contribute to the firing of $\boldsymbol{\sigma}[t_j]$ (they do not make $\boldsymbol{\sigma}[t_j]$ firing faster). Therefore, firing $\boldsymbol{\sigma}'[t_j]$ cannot cost less time than firing $\boldsymbol{\sigma}[t_j]$. ■

Lemma 4.2.6 holds also only for CF nets. Let us consider the following simple example:

Example 4.2.7. *Assume that in the non-CF net system in Fig. 4.3 the firing rate vector is $\boldsymbol{\lambda} = [1\ 0.01\ 1\ 1\ 1]^T$, the sampling period is $\Theta = 0.1$, and we want to reach a final state $\boldsymbol{m}_f = [0\ 0.5\ 0.5\ 4]^T$. $\boldsymbol{m}_f$ can be reached, for example, by using $\boldsymbol{\sigma} = [0\ 0.5\ 0\ 0\ 0]^T$ or $\boldsymbol{\sigma}' = [0\ 0.5\ 0\ 4\ 4]^T$. Although $\boldsymbol{\sigma} \leq \boldsymbol{\sigma}'$, we can verify that if we apply the ON/OFF controller using $\boldsymbol{\sigma}$ the final state is reached in 692 time steps; if we we apply the ON/OFF controller using $\boldsymbol{\sigma}'$ the final state is reached in only 673 time steps. In the case of $\boldsymbol{\sigma}$, only t_2 fires, so the time used for the firing of $\boldsymbol{\sigma}[t_2]$ determines the time to reach the final state. In the case of $\boldsymbol{\sigma}'$, transitions t_4 and t_5 also fire (an additional T-semiflow $[0\ 0\ 0\ 1\ 1]^T$ is fired). By firing t_4 we can increase the marking of p_2 (at some time instants, later this increased quantity of marking is moved back to p_4 by firing t_5), so t_2 fires faster; on the other hand, since the firing rate of t_2 is much smaller than the others, the firing of $\boldsymbol{\sigma}'[t_2]$ still determines the total time to reach $\boldsymbol{m}_f$. Therefore, the final state is reached faster in the case of firing $\boldsymbol{\sigma}'$.*

On the other hand, even for CF nets, given $\boldsymbol{\sigma} \leq \boldsymbol{\sigma}'$, $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$ and $\boldsymbol{m}_f{}' = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}'$, if $\boldsymbol{m}_f \neq \boldsymbol{m}_f{}'$, by applying the ON/OFF controller, reaching $\boldsymbol{m}_f{}'$ with $\boldsymbol{\sigma}'$ may be faster than reaching $\boldsymbol{m}_f$ with $\boldsymbol{\sigma}$.

Example 4.2.8. *Let us consider again the trivial MG (a subclass of CF) in Fig. 4.2 of Ex. 4.2.4. Now assume that $\boldsymbol{m}_0 = [1\ 1\ 0]^T$; $\boldsymbol{\lambda} = [10\ 1\ 1]^T$; and sampling period is $\Theta = 0.01$. Given $\boldsymbol{\sigma} = [0\ 0.5\ 0]^T$ and $\boldsymbol{\sigma}' = [0.5\ 0.5\ 0]^T$ ($\boldsymbol{\sigma} \leq \boldsymbol{\sigma}'$), by using the ON/OFF controller, $\boldsymbol{\sigma}$ is fired in 69 time steps ($[1\ 0.5\ 0.5]^T$ is reached) and $\boldsymbol{\sigma}'$ is fired in only 42 time steps ($[0.5\ 1\ 0.5]^T$ is reached).*

Proposition 4.2.9. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a consistent and strongly connected discrete-time continuous CF net system and $\boldsymbol{\sigma} \geq 0$ be a firing count vector driving the system*

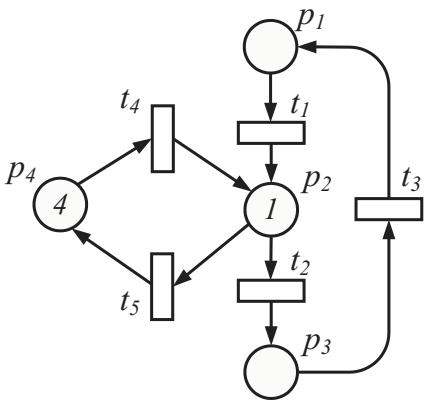


Figure 4.3: A simple non-CF net (a state machine): by using the ON/OFF controller, firing $\boldsymbol{\sigma}$ may cost more time than firing $\boldsymbol{\sigma}' \geq \boldsymbol{\sigma}$

to $\boldsymbol{m}_f > 0$*, i.e.,* $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$*. The ON/OFF controller is a minimum-time controller driving the system to* $\boldsymbol{m}_f$ *if* $\boldsymbol{\sigma}$ *is the minimal firing count vector.*

Proof: In consistent and strongly connected CF nets, there exist a unique minimal firing count vector and a unique minimal T-semiflow $\boldsymbol{x} > 0$ [93], hence for any other firing count vector $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}$ that drives the system to $\boldsymbol{m}_f$ we must have $\boldsymbol{\sigma}' = \boldsymbol{\sigma} + \alpha \cdot \boldsymbol{x} > \boldsymbol{\sigma}$, $\alpha > 0$. According to Lemma 4.2.5 and 4.2.6, the results can be derived straightforwardly. ■

Remark 4.2.10. *When the final state* $\boldsymbol{m}_f$ *has been reached by applying the ON/OFF controller, all the transitions are stopped. If* $\boldsymbol{m}_f$ *is an equilibrium point, it can be maintained by using an appropriated control* $\boldsymbol{u}_k$*, such that* $\boldsymbol{C} \cdot \boldsymbol{w}_k = 0$*.*

4.2.3 ON/OFF controller: continuous-time case

By taking the sampling period $\Theta \to 0$, the ON/OFF controller can be easily extended to the continuous time setting, the control input for transition t_j at time τ is given by:

$$\boldsymbol{u}(\tau)[t_j] = \begin{cases} 0 & \text{if } \int_0^{\tau^-} \boldsymbol{w}(\delta)[t_j]\,\mathrm{d}\delta < \boldsymbol{\sigma}[t_j] \quad (ON) \quad (a) \\ \boldsymbol{f}(\tau)[t_j] & \text{if } \int_0^{\tau^-} \boldsymbol{w}(\delta)[t_j]\,\mathrm{d}\delta = \boldsymbol{\sigma}[t_j] \quad (OFF) \quad (b) \end{cases} \tag{4.7}$$

where $\boldsymbol{\sigma}$ is the minimal firing count vector and $\boldsymbol{w}(\delta)[t_j]$ is the controlled flow of t_j at time δ; $\boldsymbol{f}(\tau)[t_j]$ is the uncontrolled flow at time τ.

It should be noticed that for continuous timed systems under infinite server semantics, once a place is marked it will take infinite time to be emptied (like the discharging of a capacitor in an electrical RC-circuit). Therefore, if there exist places that are emptied during the trajectory to $\boldsymbol{m}_f$, the final marking is reached at the limit, i.e., in infinite time. If $\boldsymbol{m}_f > 0$ and use the proposed control method, this situation does not happen.

One main advantage of the ON/OFF control strategy is its low computational complexity. Given a (minimal) firing count vector (that can be computed in polynomial time), the control actions can be obtained by solving a simple LPP (also in polynomial time) in each time step. On the other hand, the minimum-time state evolution is guaranteed.

Although the ON/OFF is only proposed for CF nets, we can also guarantee its convergence to the final state if the system is Join-Free (JF) and conservative. But now it cannot ensure a minimum-time evolution to $\boldsymbol{m}_f$ in general.

Proposition 4.2.11. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a conservative Join-Free TCPN system and $\boldsymbol{\sigma} \geq 0$ be a firing count vector diving the system to $\boldsymbol{m}_f > 0$, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. By applying the ON/OFF controller, the system state converges to $\boldsymbol{m}_f$ in finite time.*

Proof: We prove it by contradiction. Assume that the system reaches a state $\boldsymbol{m}$ and $\boldsymbol{m} \neq \boldsymbol{m}_f$. Therefore, there must exist a transition t_j that cannot reach its accumulative firing upper bound $\boldsymbol{\sigma}[t_j]$. Since the system is JF, t_j can fire if its unique input place p_i is not emptied, it implies that $\boldsymbol{m}[p_i] = 0$. For any other place p_i' such that its output transitions fire completely the firing amounts give by $\boldsymbol{\sigma}$, it holds $\boldsymbol{m}[p_i'] \leq \boldsymbol{m}_f[p_i']$ because the firing of every transition is upper bounded by $\boldsymbol{\sigma}$ and $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. Therefore, for any place $p \in P$ it holds $\boldsymbol{m}[p] \leq \boldsymbol{m}_f[p]$ and there exists at least one place $p_i \in P$, such that $\boldsymbol{m}[p_i] = 0 < \boldsymbol{m}_f[p_i]$. This contradicts the conservativeness of the net. ■

Remark 4.2.12. *Let us notice that in the standard ON/OFF controller proposed here, we do not necessarily require a positive initial marking. However, it is needed for the extended controllers for general nets that will be presented in Sections 4.4.*

4.2.4 A case study

Let us consider the net system in Fig. 4.4, which models a table factory system (taken from [93]). The system consists of several parts, including board maker, leg maker, assembler, painting line. Assume that 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; in the final marking $m_f[p_3] = m_f[p_{17}] = 0.1$, $m_f[p_4] = m_f[p_5] = 0.2$, $m_f[p_{13}] = 0.15$, and all the other places with markings equal to 0.25. The corresponding minimal firing count vector $\boldsymbol{\sigma} = [0.85\ 0.85\ 1.0\ 0.9\ 0.6\ 0.6\ 0.75\ 0.65\ 0.45\ 0.2\ 0.35\ 0.10]^T$.

Fig. 4.5 shows the stopping time instants of transitions when the ON/OFF controller is applied. After t_9 is stopped at 4.28 time units, the markings of all the places are at the final state values.

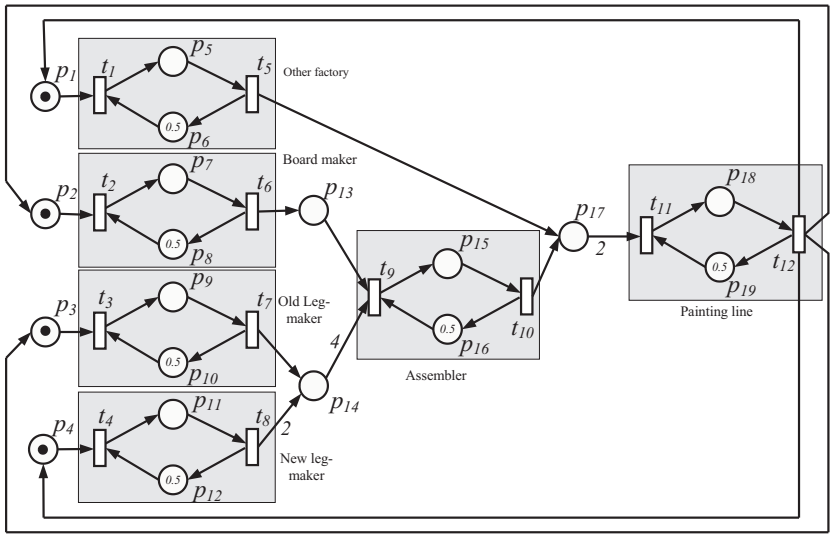


Figure 4.4: The CF net model (weighted T-system) of a table factory system. The firing rate of every transition is equal to 1.

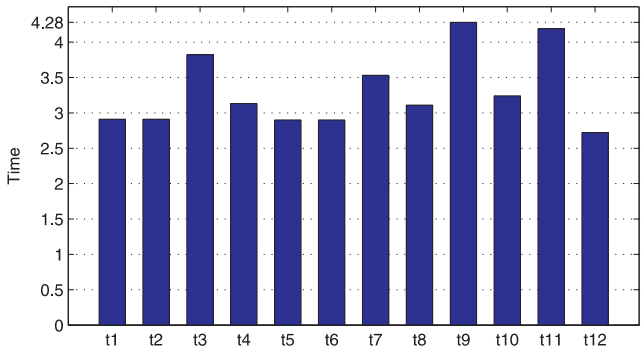


Figure 4.5: Transitions' stopping time instants obtained by applying the ON/OFF controller to the CF system in Fig. 4.4

Fig. 4.6 shows the marking trajectory of places p_3, p_{13}, p_{14} and p_{17}. For instance, the marking of place p_{17} depends 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} stops, p_{17} also reaches its final state, at 4.19 time units.

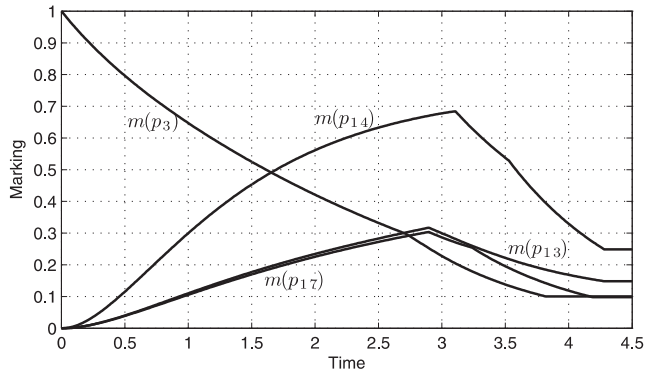


Figure 4.6: Marking trajectories of applying the ON/OFF controller to the CF system in Fig. 4.4

4.3 Drawbacks of the ON/OFF controller for general nets

In general net systems, multiple minimal firing count vectors may exist. Therefore it is not clear which one gives the minimum-time by using the ON/OFF controller (moreover, minimum-time may be with a non-minimal firing count vector). On the other hand, the convergence of the final state may not be ensured: in the case of non-CF nets, conflicts ($|p^\bullet| > 1$) may appear, thus firing faster one transition may reduce the firing of another transition, and the overall time for reaching $\boldsymbol{m}_f$ may increase, being infinity in the extreme case. The following example shows a live and bounded system, in which by applying the ON/OFF strategy, the final state cannot be reached.

Example 4.3.1. *Assume we want to drive the system in Fig.4.7 to final state* $\boldsymbol{m}_f = [0.5\ 0.5\ 0.5\ 0.5\ 0.2\ 0.4\ 1.4]^T$*, the firing rate of* t_3 *is 10, while the firing rates of other transitions are all set to 1.* $\boldsymbol{\sigma} = [0.8\ 1.3\ 0.5\ 0\ 1\ 0\ 0]^T$ *is a minimal firing count vector driving the system from* $\boldsymbol{m}_0$ *(shown in the figure) to* $\boldsymbol{m}_f$*. By using this setting and applying the ON/OFF controller,* $\boldsymbol{m}_f$ *cannot be reached and the system will be "blocked" in an intermediate marking* $\boldsymbol{m} = [1\ 0\ 0.78\ 0.22\ 0\ 0\ 2]^T$*. Notice that, this "blocking" situation is imposed by the controller. For instance, transition* t_7 *is actually enabled at* $\boldsymbol{m}$*, but the control law has forbidden its firing because* $\boldsymbol{\sigma}[t_7] = 0$*.*

One may think that deadlock-freeness is a sufficient condition for applying the ON/OFF controller to a general net system. But we should notice that, the control laws may forbid the firings of some transitions (like in Ex.4.3.1, the firing of t_7 is forbidden because $\boldsymbol{\sigma}[t_7] = 0$), bringing the system to some "blocking" situations.

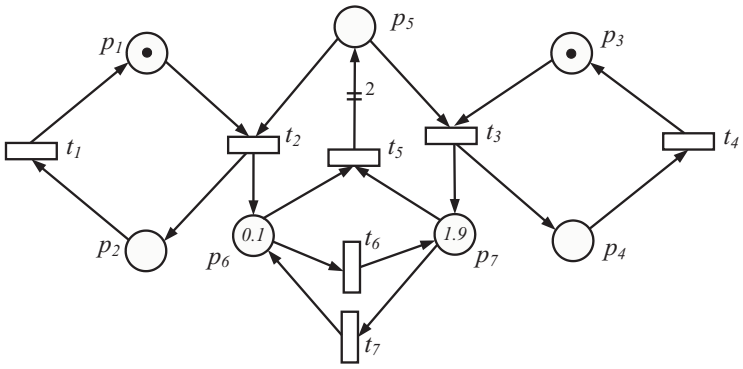


Figure 4.7: A live and bounded CPN system that the ON/OFF controller brings to a "deadlock" situation if $\lambda_3 \gg \lambda_2$

4.4 Extended ON/OFF based methods

Because the ON/OFF controller cannot be directly applied to general TCPNs, three heuristic extensions are proposed: ON/OFF+, B-ON/OFF and MPC-ON/OFF. In all the methods the convergence to the final state is always guaranteed, although we may not obtain a minimum-time state evolution. The ON/OFF+ overcomes the problem of the standard ON/OFF controller by forcing proportional firings of conflicting transitions; B-ON/OFF is proposed to handle those bad cases of applying the ON/OFF+ controller; the MPC-ON/OFF controller has higher computational complexity, but may lead to better solutions, i.e., solutions that need less time to reach the final state.

4.4.1 ON/OFF+ controller

The problem of the ON/OFF controller arises from "inappropriate" manners of solving the conflicts (e.g., in the system of Fig. 4.7, since $\lambda_3 \gg \lambda_2$, t_3 fires much faster than t_2). Two transitions t_a and t_b are in a structural conflict relation if ${}^\bullet t_a \cap {}^\bullet t_b \neq \emptyset$. The *coupled conflict* relation is its transitive closure. For example, in the net shown in Fig. 4.7, the sets of places in coupled conflict relation are $\{t_1\}$, $\{t_4\}$, $\{t_2, t_3\}$ and $\{t_5, t_6, t_7\}$. In the sequel, let us denote by T_p the set of persistent transitions (transitions that are not in any conflict relation) and T_c the set of transitions in any coupled conflict relation, $T_p \cap T_c = \emptyset, T_p \cup T_c = T$.

In order to overcome this problem, we consider a more "fair" strategy to solve the conflicts: forcing the flows of transitions that are in coupled conflict relation to be proportional to the given firing count vector. Meanwhile, for the rest of (persistent) transitions the ON/OFF strategy is applied.

The modified ON/OFF controller is shown in Algorithm 2 and we will call it ON/OFF+ controller.

The procedure of the ON/OFF+ controller is similar to the one of the standard ON/OFF, except the last constraint of LPP (4.8) in the step 3 of Algorithm 2, which

Algorithm 2 ON/OFF+ controller

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{\sigma}$, Θ

Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

1: $k \leftarrow 0$

2: **while** $\Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i \neq \boldsymbol{\sigma}$ **do**

3: Solve the following LPP:

$$
\begin{aligned}
max \quad & \mathbf{1}^T \cdot \boldsymbol{w}_k \\
\text{s.t.} \quad & \boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k \\
& \mathbf{0} \leq \Theta \cdot \boldsymbol{w}_k \leq \boldsymbol{\sigma} - \Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i \\
& \boldsymbol{w}_k[t_j] \leq \lambda_j \cdot enab(t_j, \boldsymbol{m}_k), \forall t_j \in T \\
& \boldsymbol{m}_{k+1} \geq 0 \\
& \boldsymbol{w}_k[t_a] \cdot \boldsymbol{\sigma}[t_b] = \boldsymbol{w}_k[t_b] \cdot \boldsymbol{\sigma}[t_a] \\
& \quad \forall t_a, t_b, {}^\bullet t_a \cap {}^\bullet t_b \neq \emptyset \text{ and } \boldsymbol{\sigma}[t_a] > 0, \boldsymbol{\sigma}[t_b] > 0
\end{aligned}
\tag{4.8}
$$

4: Apply $\boldsymbol{w}_k : \boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$

5: $k \leftarrow k + 1$

6: **end while**

7: **return** $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

means that, at any time step k, if transitions t_a and t_b are in conflict, the following will be forced: $\frac{\boldsymbol{w}_k[t_a]}{\boldsymbol{w}_k[t_b]} = \frac{\boldsymbol{\sigma}[t_a]}{\boldsymbol{\sigma}[t_b]}$. Since we need positive flows ($\boldsymbol{w}_k[t_b] > 0$), in the sequel we assume an initial state $\boldsymbol{m}_0 > 0$. Also Notice that, only transitions with positive values in the corresponding firing count vector should be considered.

In order to prove the convergence, we first show that by using some reduction rules, the original system with the ON/OFF+ controller is equivalent to a CF net system with a particular controller $\mathcal{A}$, i.e., the same state trajectory can be obtained. Then, we prove that controller $\mathcal{A}$ drives the CF net system to $\boldsymbol{m}_f$, implying that the ON/OFF+ controller also drives the original one to $\boldsymbol{m}_f$.

Reduction Rule. *Given a net $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$, let $T_j = \{t_1, t_2, ..., t_n\} \subseteq T$ be a set of transitions that are in coupled conflict relation. These transitions fire proportionally according to a given firing count vector $\boldsymbol{\sigma}$, i.e., for any $t_a, t_b \in T_j$, $\boldsymbol{\sigma}[t_a], \boldsymbol{\sigma}[t_b] > 0$, if t_a fires in an amount s_a, simultaneously, t_b fires in an amount s_b, such that $\frac{s_a}{s_b} = \frac{\boldsymbol{\sigma}[t_a]}{\boldsymbol{\sigma}[t_b]}$. Let $\bar{\sigma} = \sum_{t \in T_j} \boldsymbol{\sigma}[t]$, $\mathcal{N}$ is transformed to $\mathcal{N}' = \langle P, T', \boldsymbol{Pre}', \boldsymbol{Post}' \rangle$ in the following way:*

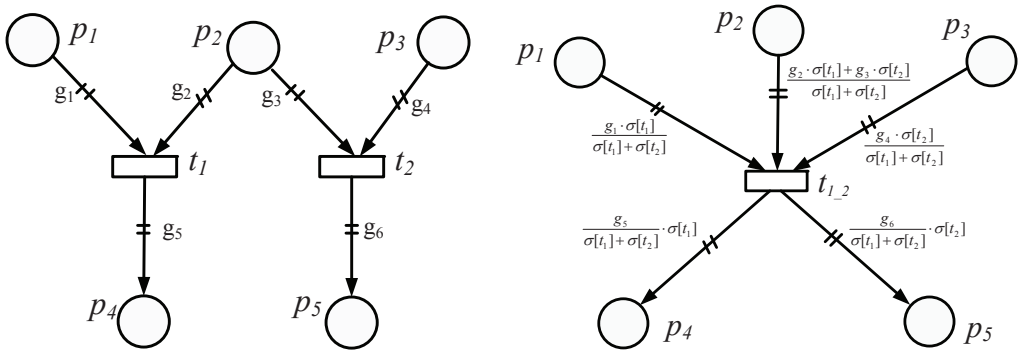
(1) $T' = T \setminus T_j$

(2) Merge T_j to a new transition t_j, $T' = T' \cup \{t_j\}$

(3) $\forall p \in {}^\bullet T_j$, $\boldsymbol{Pre}'[p, t_j] = \sum_{t \in p^\bullet} \boldsymbol{Pre}[p, t] \cdot \boldsymbol{\sigma}[t] / \bar{\sigma}$

(4) $\forall p \in T_j^{\bullet}$, $\boldsymbol{Post'}[p, t_j] = \sum_{t \in {}^{\bullet}p} \boldsymbol{Post}[p, t] \cdot \boldsymbol{\sigma}[t]/\bar{\sigma}$

Example 4.4.1. *Let* $\boldsymbol{m} > 0$ *and* $\boldsymbol{\sigma}[t_1] > 0$, $\boldsymbol{\sigma}[t_2] > 0$. *Fig. 4.8 shows how two conflicting transitions* t_1 *and* t_2 *are merged into* t_{1_2}.



(a) Original system (b) Transformed system

Figure 4.8: Dynamic reduction rule for a given $\boldsymbol{\sigma}$: merging t_1 and t_2

Proposition 4.4.2. *Let* $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, *and* $\mathcal{S}' = \langle \mathcal{N}', \boldsymbol{m}_0 \rangle$ *be the transformed system from* $\mathcal{S}$ *by merging* $T_j = \{t_1, t_2, ..., t_n\}$ *to* t_j *by using the reduction rule. If in* $\mathcal{S}$, *the transitions in* T_j *fire proportionally according to a given firing count vector* $\boldsymbol{\sigma}$, *and in* $\mathcal{S}'$, *transition* t_j *fires in an amount equal to the sum of the firing amounts of transitions in* T_j, *then the same marking is reached in* $\mathcal{S}$ *and* $\mathcal{S}'$.

Proof: It follows immediately by the definition of the reduction rule. ■

For example, consider place p_2 in Fig. 4.8, and let $s_1 = \alpha \cdot \boldsymbol{\sigma}[t_1]$, $s_2 = \alpha \cdot \boldsymbol{\sigma}[t_2]$, $\alpha > 0$. If $t_1(s_1)t_2(s_2)$ is fired in the original system, the new marking of p_2 is:

$$\boldsymbol{m}_1[p_2] = \boldsymbol{m}_0[p_2] - g_2 \cdot \alpha \cdot \boldsymbol{\sigma}[t_1] - g_3 \cdot \alpha \cdot \boldsymbol{\sigma}[t_2]$$

In the transformed system, if $t_{1\_2}(s_1 + s_2)$ is fired, the new making of p_2 is:

$$\begin{aligned} \boldsymbol{m}_1'[p_2] &= \boldsymbol{m}_0[p_2] - (s_1 + s_2) \cdot \frac{g_2 \cdot \boldsymbol{\sigma}[t_1] + g_3 \cdot \boldsymbol{\sigma}[t_2]}{\boldsymbol{\sigma}[t_1] + \boldsymbol{\sigma}[t_2]} \\ &= \boldsymbol{m}_0[p_2] - \alpha \cdot (\boldsymbol{\sigma}[t_1] + \boldsymbol{\sigma}[t_2]) \cdot \frac{g_2 \cdot \boldsymbol{\sigma}[t_1] + g_3 \cdot \boldsymbol{\sigma}[t_2]}{\boldsymbol{\sigma}[t_1] + \boldsymbol{\sigma}[t_2]} \\ &= \boldsymbol{m}_1[p_2]. \end{aligned}$$

Similarly, markings of places p_1 and p_3 are also equal in both systems.

Corollary 4.4.3. *If* $\boldsymbol{m}_f > 0$ *is reachable in* $\mathcal{S}$ *by firing* $\boldsymbol{\sigma}$ *from* $\boldsymbol{m}_0 > 0$, *then* $\boldsymbol{m}_f$ *is reachable in* $\mathcal{S}'$ *by firing* $\boldsymbol{\sigma}'$, *where:*

$$\boldsymbol{\sigma}'[t_j] = \begin{cases} \sum_{t \in T_j} \boldsymbol{\sigma}[t] & \text{if } t_j \text{ is obtained by merging a set of transitions } T_j \\ \boldsymbol{\sigma}[t_j] & \text{otherwise} \end{cases}$$

Proposition 4.4.4. *Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a discrete-time TCPN system with $\boldsymbol{m}_0 > 0$ and Θ the sampling period. Let $\boldsymbol{\sigma} \geq 0$ be a firing count vector diving the system to $\boldsymbol{m}_f > 0$, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. By applying the ON/OFF+ controller, the system state converges to $\boldsymbol{m}_f$ in finite time.*

Proof: Let $\mathcal{S}' = \langle \mathcal{N}', \boldsymbol{\lambda}', \boldsymbol{m}_0 \rangle$ be the system transformed from $\mathcal{S}$ by merging all the conflicting transitions, using the reduction rule (therefore $\mathcal{S}'$ is CF).

Assume there exists a controller $\mathcal{A}$ applied to $\mathcal{S}'$, with $\boldsymbol{w}'_k[t_j]$ the controlled flow at each time step k, such that: (1) if t_j is obtained by merging a set of transitions T_j in a coupled conflict relation, we have $\boldsymbol{w}'_k[t_j] = \sum_{t \in T_j} \boldsymbol{w}_k[t]$; (2) otherwise $\boldsymbol{w}'_k[t_j] = \boldsymbol{w}_k[t_j]$, where $\boldsymbol{w}_k[t_j]$ is the flow of transition t_j in $\mathcal{S}$ that is controlled by using the ON/OFF+ controller. Then, according to Proposition 4.4.2, the state trajectory of $\mathcal{S}'$ obtained by applying controller $\mathcal{A}$ is the same as in $\mathcal{S}$ obtained by applying the ON/OFF+ controller. Therefore it is equivalent to prove that by applying controller $\mathcal{A}$ to $\mathcal{S}'$, $\boldsymbol{m}_f$ is reached in finite time.

This controller $\mathcal{A}$ always exists, because if the firing rate of t_j in $\mathcal{S}'$, λ'_j, is large enough, case (1) can always be satisfied, by using a positive control action $\boldsymbol{u}_k[t_j]$. In particular, it is defined as:

$$\boldsymbol{u}_k[t_j] = \lambda'_j \cdot enab(t_j, \boldsymbol{m}_k) - x_k^j \tag{4.9}$$

where x_k^j is obtained by solving the LPP (4.10):

$$\begin{aligned} x_k^j = \quad & max \quad \sum_{t_d \in T_j} x_k^d \\ & \text{s.t.} \quad x_k^a \cdot \boldsymbol{\sigma}[t_b] = x_k^b \cdot \boldsymbol{\sigma}[t_a], \forall t_a, t_b \in T_j \\ & \qquad 0 \leq x_k^d \leq \lambda_d \cdot enab(t_d, \boldsymbol{m}_k), \forall t_d \in T_j \\ & \qquad \Theta \cdot \sum_{i=0}^{k} x_k^d \leq \boldsymbol{\sigma}[t_d], \forall t_d \in T_j \end{aligned} \tag{4.10}$$

where $enab(t_d, \boldsymbol{m}_k)$, $t_d \in T_j$, is the enabling degree of t_d in the original system at $\boldsymbol{m}_k$.

For case (2) we simply use the ON/OFF strategy and the same firing rate as in $\mathcal{S}$.

Finally, let us notice that $\mathcal{S}'$ is a CF net, so for sure controller $\mathcal{A}$ can drive $\mathcal{S}'$ to its final state in finite time [104], implying that by applying the ON/OFF+ controller to $\mathcal{S}$, the final state is also reached in finite time. ■

Given a firing count vector $\boldsymbol{\sigma}$, if transition t_j is a persistent one and the goal is to minimize the time spent for firing $\boldsymbol{\sigma}[t_j]$, the ON/OFF strategy is the optimal. For the transitions in conflict, the ON/OFF+ controller gives a way to handle their firings. However in general, it is just a successful heuristic (a solution always exists), but not necessarily the optimal (the minimum-time is not guaranteed).

Example 4.4.5. *Let us consider the net system in Fig. 4.9. Assume that the desired final state is $\boldsymbol{m}_f = [3.6\ 0.4\ 4\ 1.6]^T$, the firing rate vector $\boldsymbol{\lambda} = \mathbf{1}$, and the sampling*

period $\Theta = 0.2$. *One minimal firing count vector to reach* $\boldsymbol{m}_f$ *(in this case, the unique one) is* $\boldsymbol{\sigma} = [0.4\ 0\ 4\ 0]^T$. *By applying the ON/OFF+ controller, at each time step the firing flow of* t_3 *is forced to be 10 times the flow of* t_1. *For instance, at the first time step,* $[0.04\ 0\ 0.4\ 0]^T$ *is fired, reaching making* $[7.56\ 0.04\ 0.4\ 1.96]^T$, *etc. In this way,* $\boldsymbol{m}_f$ *is reached in 12 time steps. However,* $\boldsymbol{m}_f$ *could be reached in only 10 time steps (that is actually the minimum-time). In particular, at each of the first 9 time steps only* t_3 *fires, i.e.,* $[0\ 0\ 0.4\ 0]^T$ *is fired; at the last time step,* t_1 *and* t_3 *fire, i.e.,* $[0.4\ 0\ 0.4\ 0]^T$ *is fired.*

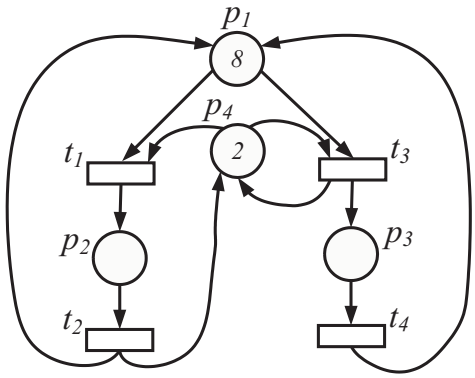


Figure 4.9: An EQ PN system with $\boldsymbol{m}_0 = [8\ 0\ 0\ 2]^T$.

Remark 4.4.6. *The results of Proposition 4.4.4 can be naturally extended to continuous-time CPNs by making the sampling period* Θ *tend to* 0.

4.4.2 Balanced ON/OFF controller (B-ON/OFF)

We can apply the ON/OFF+ controller to any TCPN system and ensure the convergence to a final state $\boldsymbol{m}_f > 0$. Extremely fast to compute, nevertheless a possible drawback of this method is the following: since a set of transitions in coupled conflict relation are forced to fire proportionally, the required number of time steps for firing $\boldsymbol{\sigma}$ is determined by the "slowest" ones. Therefore, in extreme cases, when some of these transitions have very small flows, the whole system may be slowed down.

Example 4.4.7. *Let us consider the simple (sub-)system in Fig. 4.10, assuming that* t_1, t_2 *have the same firing rate equal to* 1. *Moreover, they are forced by a given* $\boldsymbol{\sigma}$ *to fire in the same amounts. It is obvious that the flow of* t_2 *is 100 times the flow of* t_1, *but if* t_1 *and* t_2 *should fire proportionally according to* $\boldsymbol{\sigma}$, *then* t_2 *is slowed down.*

To overcome cases like that, we can fire first the "faster" transitions and block the "slower" ones for some time periods, expecting that the flows (speeds) of some of the "slower" transitions are increased, i.e., we expect somehow to "*balance*" the

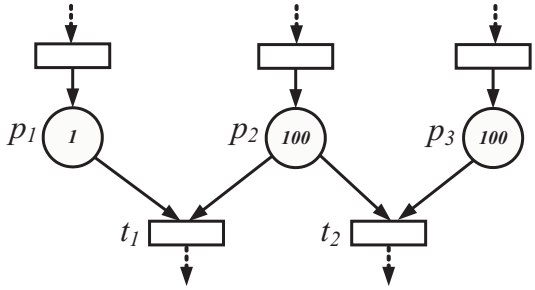


Figure 4.10: Fast transitions may be slowed down

"faster" and "slower" transitions. After that, we simply apply the pure ON/OFF+ controller until the final state is reached.

We will first show how to classify the "slower" and "faster" transitions, and then present this balancing strategy.

Assume that the system is at marking $\boldsymbol{m}$ with $\boldsymbol{w}$ its controlled flow, and let $\boldsymbol{\sigma}$ be the firing count vector that should be fired to reach $\boldsymbol{m}_f$. Then $s_j = \lceil \frac{\boldsymbol{\sigma}[t_j]}{\Theta \cdot \boldsymbol{w}[t_j]} \rceil$ can be viewed as an estimation of the number of steps that transition t_j needs to fire, assuming that t_j fires with a constant speed equal to $\boldsymbol{w}[t_j]$. For two transitions t_a and t_b, if $s_a > s_b$, it is said that t_a is "slower" than t_b.

The estimation of the number of steps for t_j at $\boldsymbol{m}_0$ is defined by:

$$s_j^0 = \left\lceil \frac{\boldsymbol{\sigma}[t_j]}{\Theta \cdot \lambda_j \cdot enab(t_j, \boldsymbol{m}_0)} \right\rceil \tag{4.11}$$

Let us consider again the system in Ex.4.4.7 and let $\boldsymbol{\sigma}[t_1] = \boldsymbol{\sigma}[t_2] = 10$, $\Theta = 0.01$. The initial estimation of the number of time steps is: $s_1^0 = \frac{10}{1 \cdot 0.01} = 1000$, $s_2^0 = \frac{10}{100 \cdot 0.01} = 10$. So transition t_1 is "slower" than transition t_2.

Based on this initial estimation, we partition any given set of transitions T_c that are in coupled conflict relation into two subsets, the "faster" ones T_1 and the "slower" ones T_2, such that:

$$\begin{cases} T_1 \cap T_2 = \emptyset, T_1 \cup T_2 = T_c \\ \forall t_a \in T_1, t_b \in T_2, s_b^0 / s_a^0 > d \\ \forall t_{a1}, t_{a2} \in T_1, s_{a1}^0 / s_{a2}^0 \leq d \end{cases} \tag{4.12}$$

where $d \geq 1$ is a design parameter used to classify "slower" and "faster" transitions.

From (4.12), the estimations of the number of time steps of the transitions in T_2 are at least d times as large as the ones of transitions in T_1. If we fire the transitions in T_1 and T_2 proportionally, transitions in T_1 may be slowed down by the ones in T_2.

Notice that, if the value of d is too large, all the transitions are put into T_1, then it is equivalent to applying the ON/OFF+ controller directly; if d is too small, most of the transitions are put into T_2 and initially blocked. For example, in the system shown in Ex. 4.4.7, because $s_1^0 / s_2^0 = 100$, for any $d < 100$ the conflicting transition set $T_c = \{t_1, t_2\}$ is partitioned to $T_1 = \{t_2\}$ and $T_2 = \{t_1\}$.

Now let us consider that the system is at time step k with marking $\boldsymbol{m}_k$, and the firing count vector $\boldsymbol{\sigma}'$ has been fired, i.e., $\boldsymbol{m}_k = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}'$. The remaining firing count vector that should be fired is $\boldsymbol{\sigma}_k = \boldsymbol{\sigma} - \boldsymbol{\sigma}' \geq 0$. The estimation of the number of steps for transition $t_j \in T_c$ at $\boldsymbol{m}_k$ is defined by:

$$s_j^k = \begin{cases} \lceil \frac{\boldsymbol{\sigma}_k[t_j]}{\Theta \cdot \boldsymbol{w}_k[t_j]} \rceil, & \text{if } t_j \in T_1 \\ \lceil \frac{\boldsymbol{\sigma}_k[t_j]}{\Theta \cdot \lambda_j \cdot enab(t_j, \boldsymbol{m}_k)} \rceil, & \text{if } t_j \in T_2 \end{cases}$$

where $\boldsymbol{w}_k[t_j]$ is the flow of transition t_j when the ON/OFF+ strategy is applied. Because the transitions in T_1 fire proportionally, for any $t_j \in T_1$, the same estimation s_j^k is obtained, denoted by h^k. For any $t_b \in T_2$, let $D_b^k = s_b^k / h^k$, which reflects the "difference" of the estimations between t_b and the "faster" transitions.

Let T_c^i, $i = 1, 2, 3, ..., l$ be the sets of transitions in coupled conflict. Algorithm 3 synthesizes the control method: for transitions in T_p, the ON/OFF strategy is always applied; for any $T_c^i = T_1^i \cup T_2^i$, those "faster" transitions in T_1^i fire proportionally using the ON/OFF+ strategy; while every "slower" transition t_b in T_2^i is blocked. Applying this strategy until t_b gets balanced with the "faster" transitions, i.e., conditions (C1) is satisfied; or until the "difference" between t_b and the "faster" transitions cannot decrease, i.e., condition (C2) is satisfied, then we move t_b to T_1^i and start to fire it using the ON/OFF+ strategy. When $T_2^i = \emptyset$, i.e., all the transitions are moved to T_1^i, it is equivalent to apply the pure ON/OFF+ controller to the system.

(C1) $D_b^k \leq d$

(C2) $D_b^k \geq D_b^{k-1}$

Now we prove the convergence of this B-ON/OFF controller to the desired final state.

Proposition 4.4.8. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a TCPN system with $\boldsymbol{m}_0 > 0$ and $\boldsymbol{\sigma} \geq 0$ be a firing count vector diving the system to $\boldsymbol{m}_f > 0$, i.e., $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. Given a parameter $d \geq 1$, by applying the B-ON/OFF controller, the system state converges to $\boldsymbol{m}_f$ in finite time.*

Proof: According to the algorithm, any set of conflicting transitions T_c^i is first divided into subset T_1^i of "faster" transitions and subset T_2^i of "slower" transitions according to the value of d. Any transition $t_b \in T_2^i$ is initially blocked and all the transitions in T_1^i are fired by using the ON/OFF+ strategy. In this way, more tokens may arrive to the input places of t_b and its flow may increase, consequently the value of D_b^k may decrease. If the value of D_b^k decreases to d then condition (C1) is satisfied; if at one moment, the value of D_b^k cannot decrease any more, then condition (C2) is satisfied. When one of conditions (C1) and (C2) is satisfied, t_b is moved from T_2^i to T_1^i and starts firing using the ON/OFF strategy.

In finite time, all the transitions in T_2^i will be moved to T_1^i , so the system is controlled by the pure ON/OFF+. Now, we only need to prove that by this moment, the system is in a state $\boldsymbol{m} > 0$ and $\boldsymbol{m}_f$ is reachable from $\boldsymbol{m}$.

Algorithm 3 B-ON/OFF Controller

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{\sigma}$, Θ, d , T_p, T_c^i, $i = 1, 2, 3..., l$
Output: $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots$

Partition every T_c^i into T_1^i and T_2^i, $i = 1, 2, ..., l$
$k \leftarrow 0$
while $\Theta \cdot \sum_{i=0}^{k-1} \boldsymbol{w}_i \neq \boldsymbol{\sigma}$ **do**
  Obtain $\boldsymbol{w}_k[t_j]$ for any $t_j \in T_p$: applying the ON/OFF strategy
  for $i = 1$ **to** l **do**
    For any $t_j \in T_2^i$: $\boldsymbol{w}_k[t_j] \leftarrow 0$
    Obtain $\boldsymbol{w}_k[t_j]$ for any $t_j \in T_1^i$: applying the ON/OFF+ strategy
  end for
  Apply $\boldsymbol{w}_k$: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$
  $\boldsymbol{\sigma}_{k+1} \leftarrow \boldsymbol{\sigma} - \Theta \cdot \sum_{i=0}^{k} \boldsymbol{w}_i$
  for $i = 1$ **to** l **do**
    if $T_2^i \neq \emptyset$ **then**
      Compute $\boldsymbol{w}_{k+1}[t_a], t_a \in T_1^i$
      $h^{k+1} \leftarrow \boldsymbol{\sigma}_{k+1}[t_a]/(\Theta \cdot \boldsymbol{w}_{k+1}[t_a])$
      for each $t_b \in T_2^i$ **do**
        $s_b^{k+1} \leftarrow \boldsymbol{\sigma}_{k+1}[t_b]/(\Theta \cdot \lambda_b \cdot enab(t_b, \boldsymbol{m}_{k+1}))$
        $D_b^{k+1} \leftarrow s_b^{k+1}/h^{k+1}$
        if $D_b^{k+1} \leq d$ **or** $D_b^{k+1} \geq D_b^k$ **then**
          $T_1^i \leftarrow T_1^i \cup \{t_b\}$
          $T_2^i \leftarrow T_2^i \setminus \{t_b\}$
        end if
      end for
    end if
  end for
  $k \leftarrow k + 1$
end while
return $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots$

Since the accumulative firing counts of transitions are upper bounded by $\boldsymbol{\sigma}$, then we have $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}'$, $0 \leq \boldsymbol{\sigma}' \leq \boldsymbol{\sigma}$. Because $\boldsymbol{m}_0 > 0$, in a finite time $\boldsymbol{m} > 0$. Since $\boldsymbol{\sigma} - \boldsymbol{\sigma}' \geq 0$ and $\boldsymbol{m}_f = \boldsymbol{m} + \boldsymbol{C} \cdot (\boldsymbol{\sigma} - \boldsymbol{\sigma}') > 0$, $\boldsymbol{m}_f$ is reachable from $\boldsymbol{m}$ ([49]). Therefore, the final state can be reached in finite time. ■

Remark 4.4.9. *The B-ON/OFF controller is computationally more expensive than the ON/OFF+ controller, because an estimation of the number of time steps has to be computed at each iteration. However, the B-ON/OFF strategy may significantly decrease the time of reaching the final state if the flows of conflicting transitions are of different orders of magnitude. The choice of the design parameter* d *also influences*

the performance of Algorithm 3. In particular, we suggest using a small d*, because if* d *is too large, the controller is not "sensitive" to the difference between "faster" and "slower" transitions, thus it is similar to applying the ON/OFF+ strategy. In the example of Section 4.4.5, we use* $d \leq 10$ *and reasonable results are obtained.*

4.4.3 MPC-ON/OFF controller

Both the ON/OFF+ and B-ON/OFF controllers are using some "greedy strategies" to fire transitions, they solve the conflict based on the flows and the required firing counts in a current time step, without a "careful looking at the future". In this section, we combine the ON/OFF strategy with Model Predictive Control (MPC), obtaining the MPC-ON/OFF controller.

MPC has been widely applied in the industry for controlling complex dynamic systems. By solving a discrete-time optimal control problem over a given time horizon, an optimal open-loop control input sequence is obtained and the first one is applied. Then at the next time step, a new optimal control problem is solved. In [64], the MPC scheme is applied to the control of TCPNs, by solving the following optimization problem (3.5) at each time step, with cost function $J(\boldsymbol{m}_k, N)$ (3.6):

MPC is usually used for optimizing trajectories subject to certain constraints. In our problem, the aim is to reach $\boldsymbol{m}_f$ as soon as possible, i.e., minimizing the time. Although it is difficult to obtain a minimum-time control by using a MPC approach, we will consider this method for transitions in conflicts while for the others we use a similar ON/OFF strategy. We may obtain smaller number of time steps than those of the ON/OFF+ or B-ON/OFF controller, particularly with large time horizon N (even if an improvement is not guaranteed by using a larger N), what means with higher computational complexity.

The MPC-ON/OFF controller is synthesized in Algorithm 4.

Algorithm 4 MPC-ON/OFF controller

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{\sigma}$, Θ, $\boldsymbol{Q}$, $\boldsymbol{R}$, N ϵ, ζ
Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

1: $k \leftarrow 0$
2: $\boldsymbol{\sigma}_k \leftarrow \boldsymbol{\sigma}$
3: **while** $\boldsymbol{m}_k \neq \boldsymbol{m}_f$ **do**
4: Solve problem (4.13)
5: Apply $\boldsymbol{w}_k : \boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$
6: $\boldsymbol{\sigma}_{k+1} \leftarrow \boldsymbol{\sigma}_k - \Theta \cdot \boldsymbol{w}_k$
7: $k \leftarrow k + 1$
8: **end while**
9: return $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

The problem that should be solved at each time step k is:

$$\begin{aligned}
\min \quad & J(\boldsymbol{m}_k, N) \\
s.t.: \quad & \boldsymbol{m}_{k+j+1} = \boldsymbol{m}_{k+j} + \Theta \cdot C \cdot \boldsymbol{w}_{k+j}, j = 0, ..., N-1 && \text{(4.13a)} \\
& \boldsymbol{G} \cdot \begin{bmatrix} \boldsymbol{w}_{k+j} \\ \boldsymbol{m}_{k+j} \end{bmatrix} \leq 0, j = 0, ..., N-1 && \text{(4.13b)} \\
& \boldsymbol{w}_{k+j} \geq 0, j = 0, ..., N-1 && \text{(4.13c)} \\
& \Theta \cdot \sum_{j=0}^{N-1} \boldsymbol{w}_{k+j} \leq \boldsymbol{\sigma}_k && \text{(4.13d)} \\
& \boldsymbol{m}_{k+1} \geq \mathbf{1} \cdot \epsilon && \text{(4.13e)} \\
& \mathbf{1}^T \cdot \boldsymbol{w}_k \geq \zeta && \text{(4.13f)}
\end{aligned}$$

where ϵ and ζ are sufficient small positive numbers and $\boldsymbol{\sigma}_k$ is the remaining firing count vector that should be fired. Constraint $\boldsymbol{m}_{k+1} \geq \mathbf{1} \cdot \epsilon$ (4.13e) ensures that the system only evolves inside an interior region of the reachability space; in order to include $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$ in that region, it should hold $\boldsymbol{m}_f \geq \mathbf{1} \cdot \epsilon$ and $\boldsymbol{m}_0 \geq \mathbf{1} \cdot \epsilon$. Constraint $\mathbf{1}^T \cdot \boldsymbol{w}_k \geq \zeta$ (4.13f) forces a non-zero flow in the first predictive step. For our specific problem, we use the following assumptions:

(A1) $\boldsymbol{m}_0, \boldsymbol{m}_f > 0$.

(A2) $\boldsymbol{Q} \in \mathbb{R}^{|P|} \geq 0$ are positive definite matrices.

(A3) $\boldsymbol{R} \in \mathbb{R}_{\geq 0}^{|T|}$ is a diagonal matrix, such that if $t_j \in T_p$, $\boldsymbol{R}[j,j] > 0$, otherwise $\boldsymbol{R}[j,j] = 0$.

We define the cost function as:

$$J(\boldsymbol{m}_k, N) = \sum_{j=0}^{N} [(\boldsymbol{m}_{k+j} - \boldsymbol{m}_f)' \cdot \boldsymbol{Q} \cdot (\boldsymbol{m}_{k+j} - \boldsymbol{m}_f)] - \boldsymbol{w}_k' \cdot \boldsymbol{R} \cdot \boldsymbol{w}_k$$

By means of the item $-\boldsymbol{w}_k' \cdot \boldsymbol{R} \cdot \boldsymbol{w}_k$ in the cost function and choosing large values for $\boldsymbol{R}[j,j], t_j \in T_p$, we try to fire the persistent transitions as fast as possible, similarly to applying the ON/OFF strategy. Now, we will prove that the asymptotic stability holds.

Proposition 4.4.10. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a TCPN system with $\boldsymbol{m}_0 > 0$. Let $\boldsymbol{m}_f > 0$ be a reachable final marking, such that $\boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. Assume that the system is controlled by using the MPC-ON/OFF controller shown in Algorithm 4, and assumptions (A1)–(A3) are satisfied. Then the closed-loop system is asymptotically stable.*

Proof: We will define a Lyapunov function and prove that it is strictly decreasing.

Let $V(\boldsymbol{m}_k) = \mathbf{1}^T \cdot (\boldsymbol{\sigma} - \Theta \cdot \sum_{i=0}^{k} \boldsymbol{w}_i)$, where $\boldsymbol{w}_k$ is the controlled flow at time step k and Θ is the sampling period. According to constraint (4.13d), the accumulative

firing count is upper bounded by $\boldsymbol{\sigma}$. Therefore, $V(\boldsymbol{m}_k) \geq \mathbf{0}$ and $V(\boldsymbol{m}_k) \neq 0$ until $\boldsymbol{\sigma} = \Theta \cdot \sum_{i=0}^{k} \boldsymbol{w}_i$, i.e., until $\boldsymbol{m}_k = \boldsymbol{m}_f$. Now we need to prove that $V(\boldsymbol{m}_k)$ is strictly decreasing, and it is equivalent to prove that $\boldsymbol{w}_k \neq 0$ until $\boldsymbol{\sigma}$ is reached. Considering the last constraint (4.13f), we only need to prove that problem (4.13) is feasible until $\boldsymbol{m}_f$ is reached.

Assume that the system is at time step k with marking $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, according to constraint (4.13e), we have $\boldsymbol{m}_k > 0$. Let us denote by $\boldsymbol{\sigma}'$ the firing count vector that has been fired. It is clear that $\boldsymbol{\sigma}' \leq \boldsymbol{\sigma}$, therefore, $\boldsymbol{\sigma} - \boldsymbol{\sigma}' \geq 0$ and:

$$\boldsymbol{m}_f = \boldsymbol{m}_k + \boldsymbol{C} \cdot (\boldsymbol{\sigma} - \boldsymbol{\sigma}') > 0 \tag{4.14}$$

so $\boldsymbol{m}_f$ is reachable from $\boldsymbol{m}_k > 0$ [49]. Since the net is consistent, the marking of the system is able to move from $\boldsymbol{m}_k$ in any direction (may be a small movement) inside the reachability space. Therefore, if ζ is small enough, we can always find a solution of problem (4.13) in which $\boldsymbol{m}_{k+1}$ is, for example, on the straight line from $\boldsymbol{m}_k$ to $\boldsymbol{m}_f$.

■

4.4.4 Initial Comparisons

In order to have a good "guess" of selecting the most appropriate technique for a given system, several qualitative properties may be taken into account. Table 4.1 shows some qualitative characteristics of the already mentioned control methods. Apart from the methods proposed in this Chapter, another heuristics proposed in [5] for minimum-time control of TCPNs is also included in the comparison.

Table 4.1: Qualitative characteristics of several control methods that all ensure the stability (assuming $\boldsymbol{m}_0 > 0$, $\boldsymbol{m}_f > 0$).

Methods	**Subclass**	**Computational issues**	**Optimizing index**
Approaching minimum-time [5]	All	A BPP for each intermediate state	Heuristic Min. Time
ON/OFF	CF	a LPP in each time step	Min. time
ON-OFF+	All	a LPP in each time step	Heuristic Min. Time
B-ON/OFF	All	a LPP in each time step	Heuristic Min. Time
MPC-ON/OFF	All	a QPP in each time step	Heuristic Min. Time

The ON/OFF controller is particularly suitable for the minimum-time control of CF nets, while all the other methods can be applied to general net systems. For the ON/OFF, ON/OFF+ and B-ON/OFF controllers, in each step only a LPP

needs to be solved, therefore those methods have very low computational complexity. Nevertheless, for the MPC-ON/OFF controller, the number of variables also depends on the time horizon N, being computationally expensive if N is large. The approaching minimum-time controller [5] also has high computational complexity, since bilinear programming problems (BPPs) have to be solved when intermediate states are added to the trajectory for decreasing the duration of the evolution.

4.4.5 A case study

In this section, we apply different control methods to the CPN model of an assembly system using different settings. The simulations are performed on a PC with Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz, 3.24GB of RAM. More case studies are in Chapter 8.

The system model in Fig. 4.11 represents an assembly system. There are two kinds of input raw materials stored in p_1 and p_2. The material A, B are first processed by $Proc\_A1$, then the obtained semi-products are further processed by $Proc\_A2$ and $Proc\_A3$. In the other processing line, material B is sequentially processed by $Proc\_B1$ and $Proc\_B2$. Then final produces are obtained after assembling all the semi-products.

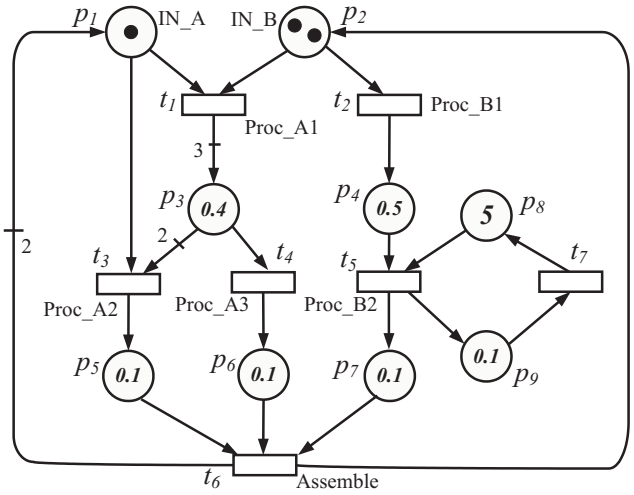


Figure 4.11: The TCPN model of an assembly system.

It is assumed that the firing rate of t_2 is 4, while for the other transitions, they are equal to 1. The simulations are performed under four different settings, in which setting **s**.1) and setting **s**.2) have different initial and final states, but the same firing count vector is used; setting **s**.3) uses the same initial marking as in **s**.1), but a different final state (therefore a different firing count vector) is considered.; the difference between setting **s**.3) and setting **s**.4) is that, for the places with markings smaller than 1 in **s**.3), their markings are increased by 1 in **s**.4), while the same firing count vectors are used in both cases.

s.1) $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$;

s.2) $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.001\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.301\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$;

s.3) $\Theta = 0.1$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 3\ 2.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$.

s.4) $\Theta = 0.02$, $\boldsymbol{m}_0 = [1\ 2\ 1.4\ 1.5\ 1.1\ 1.1\ 1.1\ 5\ 1.1]^T$, $\boldsymbol{m}_f =[0.6\ 1.8\ 1.7\ 1.2\ 1.2\ 1.5\ 1.3\ 3\ 3.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$.

The simulation results are shown in Table A.1–A.4 in the Appendix (for the B-ON/OFF controller, smaller numbers of time steps are usually obtained with smaller values of d. In the case that the numbers of time steps of using the ON/OFF+ controller cannot be reduced by using the B-ON/OFF controller, the result is not sensitive to d. For the MPC-ON/OFF controller, although in some cases the numbers of time steps are not very sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$, we suggest to use larger weights on matrix $\boldsymbol{R}$ and smaller weights on matrix $\boldsymbol{Q}$; we may slightly reduce the time steps with larger N, but at the same time, the computational costs grows fast with respect to N). In Table 4.2, we summarize the smallest numbers of time steps that obtained by using different control methods, and the corresponding parameters.

From the aspect of the number of time steps spent to reach the final state, the B-ON/OFF controller gives the best result in most of the cases (it is the best for setting **s**.1) (Table A.1) and setting **s**.2) (Table A.2), and close to the best in setting **s**.3) (Table A.3) and setting **s**.4) (Table A.4); usually smaller d lead to smaller numbers of time steps. The ON/OFF+ controller also gives quite small number of time steps, except in setting **s**.2). This is because that there are four transitions, t_1, t_2, t_3 and t_4, in coupled conflict relation and in setting setting $\boldsymbol{s}$.2), the initial marking of place p_3 is much smaller than those of p_1 and p_2, therefore the flows of t_3 and t_4 are much smaller than those of t_1 and t_2. As we have discussed in previous sections, in the case of conflicting transitions with very different flows, the overall system may be highly slowed down by applying ON/OFF+ controller. The approaching minimum-time controller does not give smaller numbers of time steps comparing with the other controllers, except in setting **s**.4). The reason is what we have already mentioned before: the performance (with respect to the time) of the approaching minimum-time controller highly depends on the initial state of the system, if there exist places with very small initial markings, the time to reach the final state could be large. Regarding the MPC-ON/OFF controller, the numbers of time steps are not far from the best among other control methods, even the best in setting **s**.3). We can also observe that, usually with larger time horizon N we could obtain smaller number of time steps (in setting **s**.2), **s**.3) and **s**.4)); but it is not guaranteed, for example the smallest time step in setting **s**.4) is obtained with $N = 1$. On the other hand, with larger weights to the matrix R (i.e., putting more

Table 4.2: The smallest numbers of time steps of using different control methods among different parameters (if exist), derived from Table A.1–A.4. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

	Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
	appro. min-time	101	847	
	ON/OFF+	94	41	
s.1)	B-ON/OFF	91	136	$d = 2$ (or 1)
	MPC-ON/OFF	91	955	$N = 1$, r =1000, $q = 1$ (or 100, 1000)
	MPC-ON/OFF	91	50,784	$N = 5$, $r = 1000$, $q = 1$
	appro. min-time	176	465	
s.2)	ON/OFF+	954	410	
	B-ON/OFF	132	192	$d = 2$ (or 1)
	MPC-ON/OFF	145	16,240	$N = 5$, $r = 1000$, $q = 1$
	appro. min-time	94	352	
s.3)	ON/OFF+	76	34	
	B-ON/OFF	76	115	d =20 (or 15, 10)
	MPC-ON/OFF	75	52,803	$N = 10$, r=1000, q=1000
	appro. min-time	122	2,546	
s.4)	ON/OFF+	126	55	
	B-ON/OFF	126	195	$d = 20$ (or 15, 10)
	MPC-ON/OFF	125	90,781	$N = 10$, r=1000, q=1

weight on the flow of persistent transitions), we often obtain a smaller number of the time steps.

From the computational costs point of view, the ON/OFF+ controller gives lowest consumed CPU time, except in setting **s**.2). The B-ON/OFF controller also has very low computational costs, slightly higher than that of the ON/OFF+ controller because an estimation of number of time steps should be computed. The approaching minimum-time controller usually costs more CPU time to compute the control low than the ON/OFF+ and B-ON/OFF controller, because a BPP problem needs to be solved whenever an intermediate point is added to the trajectory to improve the time. The most computationally expensive method here is the MPC-ON/OFF controller: a QPP should be solved in each time steps and with larger N its computational costs increase fast.

Notice that we have shown the results of different methods for a particular example with different settings, however, the number of time steps required for reaching the final state may depend on many variables, such as net structures, firing rates, initial/final state. More comparisons using different examples are presented in Chapter 8.

4.5 Computation of minimum-time control laws

Assume that by applying one of the ON/OFF (based) controllers, we reach the final state in h steps. For a general PN system, usually it does not give a minimum-time control law (see Ex. 4.4.5 for an example). However, after the application of the ON/OFF (based) method and reaching the final state in h steps, one may be interested in computing (or knowing) the minimum number of time steps. This can be computed by applying Algorithm 5.

The idea of the algorithm is the following: because we already know that $\boldsymbol{m}_f$ can be reached in h steps, now we verify if $\boldsymbol{m}_f$ can also be reached in $k = h-1$ steps, by solving LPP (4.15). If a control law is found, then we decrease k again and check if $\boldsymbol{m}_f$ can still be reached; repeat this process until we find a k such that $\boldsymbol{m}_f$ cannot be reached, then the minimum number of time steps to reach $\boldsymbol{m}_f$ is $k+1$. Instead of a sequential decreasing of k we may apply different approaches (e.g., *binary search* algorithms) to "search" for the minimum-time control laws. However, the problem is that the computational complexity of this kind of method may become intractable in practice when the system is large or h is very large.

Algorithm 5 Computation of minimum-time control laws

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, Θ

Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

1: Apply one of the ON/OFF based controllers
2: $\boldsymbol{W} = \{\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{h-1}\}$, $\boldsymbol{W}_{last} = \emptyset$
3: $k = h$
4: **repeat**
5: $k = k - 1$
6: Solve the following problem:

$$\begin{array}{ll} min & Z = ||\boldsymbol{m}_k - \boldsymbol{m}_f||_1 \\ \text{s.t.} & \boldsymbol{m}_{i+1} = \boldsymbol{m}_i + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_i,\ i = 0, 1, ..., k-1 \\ & \boldsymbol{w}_i[t_j] \leq \lambda_j \cdot enab(t_j, \boldsymbol{m}_i), \forall t_j \in T,\ i = 0, ..., k-1 \\ & \boldsymbol{w}_i \geq 0,\ i = 0, ..., k-1 \end{array} \tag{4.15}$$

7: **if** $Z = 0$ **then**
8: $\boldsymbol{W}_{last} = \boldsymbol{W}$
9: $\boldsymbol{W} = \{\boldsymbol{w}_0, \boldsymbol{w}_1 ... \ \boldsymbol{w}_{k-1}\}$
10: **end if**
11: **until** $Z \neq 0$
12: **return** $\boldsymbol{W}$

Remark 4.5.1. *In problem 4.15, the minimization of* $||\boldsymbol{m}_k - \boldsymbol{m}_f||_1$ *can be turned into a LPP by introducing a new variable* $\boldsymbol{v}$ *with constraints* $\boldsymbol{v} \geq \boldsymbol{m}_k - \boldsymbol{m}_f$, $\boldsymbol{v} \geq -(\boldsymbol{m}_k - \boldsymbol{m}_f)$, *then solving the problem by minimizing* $\mathbf{1}^T \cdot \boldsymbol{v}$.

4.6 Conclusions

The time spent on the trajectory and the computational complexity of control synthesis are naturally important for the targeting control problem that we address in this thesis. In this chapter we develop several controllers for TCPNs under infinite server semantics, based on the ON/OFF strategy, which frequently arises in minimum-time problem. The basic idea is to fire transitions as fast as possible until a upper bound is reached. This upper bound is specified by a given firing count vector that brings the system to its desired final state. The final state, if an equilibrium point, can be maintained by using proper control inputs. We first prove that for CF net systems, the standard ON/OFF controller ensures minimum-time evolution. For a general net system, we only obtain heuristic minimum-time by using one of the extended ON/OFF methods—additional techniques for solving conflicts are used. Low computational complexity is a main advantage of our methods. We can see from Table 4.2 that, the ON/OFF+ and B-ON/OFF controllers have much lower computational costs than the approaching minimum-time controller. By applying the proposed methods, we obtain a reasonable trade-off on quality vs computational complexity: relatively small numbers of time steps for reaching the final state and acceptable computational costs.

Chapter 5

Decentralized Control of CF nets: ON/OFF Based Methods

In this chapter, decentralized methods for the target marking control problem of TCPNs are studied. Here, we assume nets to be Choice-Free; they are cut into disjoint subnets through a set of buffer places. Due to the disconnection of subsystems, different behaviors may appear. In order to overcome this problem, we propose two reduction rules to obtain an abstraction of the missing part of each disconnected subsystem. The abstractions are then used to complement the subsystems; in this way, the behaviors (firing sequences) of the original system are preserved. Algorithms are proposed to make an agreement among those local control laws computed separately in complemented subsystems, because they may be not globally admissible considering the states of the buffer places. After that, the minimum-time ON/OFF controller presented in Section 4 can be implemented independently in subsystems.

5.1 Problem definitions

Let us consider a large scale discrete event dynamic system, e.g., a complex transportation system connecting cities from different countries. The distributed physical deployment of the system often makes it impossible to implement a centralized controller that knows the detailed structures and the current states of all subsystems. A more practical way to proceed is to have local controllers allocated to each subsystem, which is the essence of decentralized control. The intersections among neighboring subsystems (in our case, modelled by places) play an important role in facilitating the interaction and communication between neighboring subsystems.

In this chapter, this method is extended to Choice-Free (CF) nets. It is assumed that the original system modelled by a CPN is cut into disconnected subsystems by sets of places (buffers). The addressed problem is to compute the control law to drive the system from an initial state to a desired final one, in a decentralized way: local controllers first compute control laws separately, then based on the local control laws, a globally admissible one is derived without knowing the detailed structures of subsystems. There are two main problems arising in this process: 1) disconnected subsystems can exhibit different behaviors from the original ones, e.g., properties like liveness or boundedness in the original system may not be preserved; and 2) since the buffer places are essentially shared by more than one subsystem, there must be an agreement among the neighboring local controllers. The first problem can be overcome by complementing the subsystems with an *abstraction* of the parts that are missing. For this purpose, two reduction rules are proposed to substitute the "missing parts" by a set of places. For the second problem, a simple *coordinator controller* is introduced. Two important characteristics of the proposed method are:

- The *coordinator* does not know the detailed structures of subsystems, but only the interface transitions.
- *Local controllers* only send limited information—the firing count vector and the minimal T-semiflow—to the coordinator.

Based on the limited information, algorithms are proposed to reach an agreement. After a globally admissible control laws is obtained, simple ON-OFF controllers (presented in Chapter 4) are applied. They bring the system to the desired final state in minimum-time. The sketch of the system structure is shown in Fig. 5.1.

5.2 Related work

In the context of decentralized control on discrete PNs, some approaches have been proposed. For example, in [40, 10], decentralized supervisory control was addressed. These works focused on enforcing states to satisfy certain constraints (specifications). Contribution [7] addressed the problem of driving the system from an initial marking to a given set of desired markings, by means of adding some control places; it did not

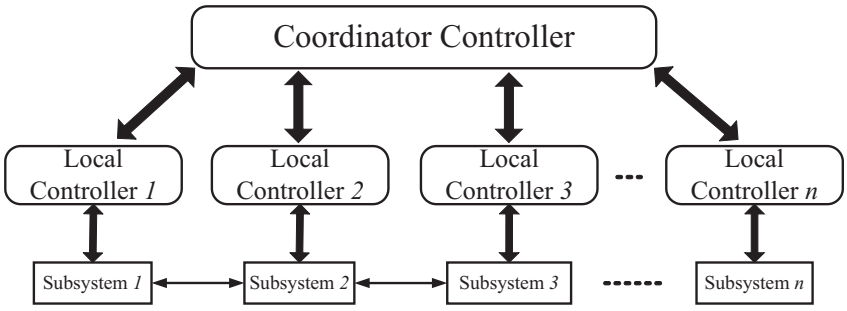


Figure 5.1: System Structures

discuss how the set of desired markings should be defined, and the control structural (the control places) highly depended on these markings.

In this chapter, we will address the decentralized target marking control problem of continuous PNs. This problem has been considered in, for example, [4, 102]. Contribution [4] considers continuous models composed by several subsystems that communicate through buffers (modelled by places). By executing the proposed algorithm iteratively in each subsystem, their respective target markings are reached and then maintained. This work contains two significant improvements with respect to [4]. Firstly, we do not assume that subsystems are strongly connected. Secondly, a globally admissible control law is achieved by a simple coordinator, therefore the iterative process executed in subsystems is not needed anymore. The method proposed in [102] considers subsystems of more general structures, where an *affine control* is applied to each subsystem, driving the system to a positive defined final state. One important difference of our method with respect to [102] is related to the communication strategy: while in [102] the coordinator needs to exchange information with subsystems during each time step, in this work, once the agreement is achieved, all the subsystems work independently, so no communication is necessary. On the other hand, the method proposed here addresses minimum-time evolution to the final state, but the affine control used in [102] does not *directly* consider any optimizing index and it is not designed for the minimum-time control.

There exist different ways to partition a large scale system into subsystems. This may be done by partitioning the sets of places and transitions as in [40, 10], or by explicitly cutting through a set of places [4, 102] or transitions [7]. In this work, subsystems are first obtained by cutting through a set of (buffer) places, then an abstraction and complementing process is applied. The obtained complemented subsystems have identical firing sequences to those of the original system. Thus, they can also be used to solve other interesting problems, for example, as in [75], for throughput approximations.

5.3 Decentralized control of CF nets

5.3.1 Cutting the system

Here the structural cutting method developed in [75] is extended to CF nets. In order to simplify the notation, we assume that the system is cut into two parts. This is not a limitation, since each part can be further divided into two more parts.

Definition 5.3.1. *Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a strongly connected CF net system, where $\mathcal{N} = \langle P \cup B, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$. B is said to be a* cut *if there exist two subnets $\mathcal{N}^i = \langle P^i, T^i, \boldsymbol{Pre}^i, \boldsymbol{Post}^i \rangle$, $i = 1, 2$, such that:*

(1) $T^1 \cup T^2 = T$, $T^1 \cap T^2 = \emptyset$

(2) $P^1 \cup P^2 = P$, $P^1 \cap P^2 = \emptyset$

(3) $P^1 \cup B = {}^\bullet T^1 \cup T^{1\bullet}$, $P^2 \cup B = {}^\bullet T^2 \cup T^{2\bullet}$

(4) $T^1 = {}^\bullet P^1 \cup P^{1\bullet}$, $T^2 = {}^\bullet P^2 \cup P^{2\bullet}$

where $U = {}^\bullet B \cup B^\bullet$ is said to be the interface, which is partitioned into U^1, U^2, such that $U^1 \cup U^2 = U$, $U^i = T^i \cap U$.

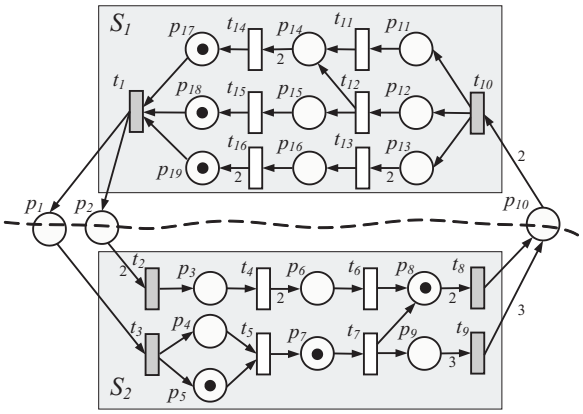
Example 5.3.2. *Fig. 5.2(a) shows a CF net system. The set of places $B = \{p_1, p_2, p_{10}\}$ is a cut decomposing the original system into two subsystems, $\mathcal{S}^1$ and $\mathcal{S}^2$, where the interface transitions are $U^1 = \{t_1, t_{10}\}$ and $U^2 = \{t_2, t_3, t_8, t_9\}$.*

5.3.2 Reduction rules

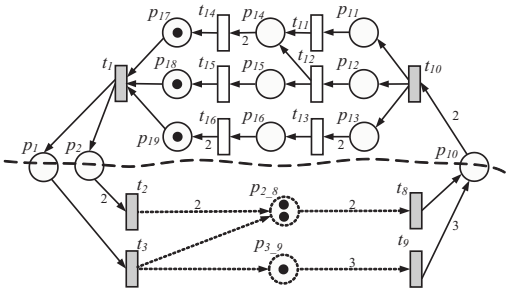
Due to the cut, different behaviors can be introduced, because subsystems become disconnected from the remaining parts. For instance, the net system in Fig.5.2(a) is live and bounded. After cutting by $B = \{p_1, p_2, p_{10}\}$, both obtained subsystems $\mathcal{S}^1$ and $\mathcal{S}^2$ become unbounded. A solution to this problem is to build an *abstraction* of the missing parts and use it to complement the disconnected subsystem.

In this section, we propose two reduction rules to obtain the *abstractions* of subsystems, in particular, the paths between interface transitions are reduced to a set of places, but no transitions. In the sequel of this section, net systems are assumed to be lim-live and bounded (in CF nets, these assumptions imply consistency and conservativeness, and if the net is connected they also imply strong connectedness). Let us first recall the concepts *gains* and *weighted markings* that we use in developing the rules.

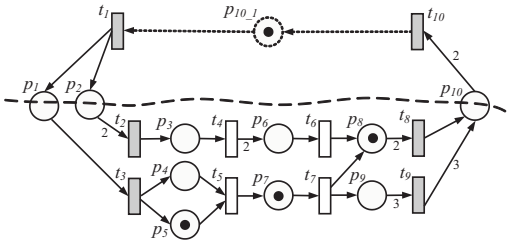
The gain of a directed path was introduced in [92] for Weighted T-system. It represents the mean firing ratio between the last transition and the first one in the path. It can be naturally extended to CF net systems:



(a)



(b) $\mathcal{CS}^1$



(c) $\mathcal{CS}^2$

Figure 5.2: (a) A live and bounded CF net system and a cut $B = \{p_1, p_2, p_{10}\}$; (b) complemented subsystem $\mathcal{CS}^1$; (c) complemented subsystem $\mathcal{CS}^2$

Definition 5.3.3. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a CF net system, and $\pi = \{t_0, p_1, t_1, p_2, ..., p_n, t_n\}$ be a directed path in $\mathcal{N}$ from transition t_0 to t_n. The gain of π is:*

$$G(\pi) = \prod_{i=1}^{n} \frac{\boldsymbol{Post}(p_i, t_{i-1})}{\boldsymbol{Pre}(p_i, t_i)}$$

The weighted marking $M(\pi, \boldsymbol{m})$ of a path π under marking $\boldsymbol{m}$ in a CF net system is the natural extension of the sum of tokens of paths in MGs.

Definition 5.3.4. *Let $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a CF net system, and $\pi = \{t_0, p_1, t_1, p_2, ..., p_n, t_n\}$ be a directed path in $\mathcal{N}$ from transition t_0 to t_n. The weighted marking of π under marking $\boldsymbol{m}$ is:*

$$M(\pi, \boldsymbol{m}) = \sum_{i=1}^{n} \left(\frac{\boldsymbol{m}[p_i]}{\boldsymbol{Post}(p_i, t_{i-1})} \prod_{j=1}^{i-1} \frac{\boldsymbol{Pre}(p_j, t_j)}{\boldsymbol{Post}(p_j, t_{j-1})} \right)$$

Let t_{in} and t_{out} (t_0 and t_n in the former definition) be the first and last transitions of π, $M(\pi, \boldsymbol{m})$ can be interpreted as the number of firings t_{in} is required to fire to

reach $\boldsymbol{m}$, in the case that π is initially empty. It can be deduced that, starting from $\boldsymbol{m}$, if all the intermediate transitions between t_{in} and t_{out} fire with the maximal amounts, the enabling degree of t_{out} becomes $G(\pi) \cdot M(\pi, \boldsymbol{m})$.

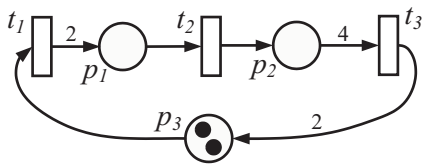


Figure 5.3: A simple CF net system with $\boldsymbol{m}_0 = [0\ 0\ 2]^T$

Example 5.3.5. *Let us consider the CF net system in Fig. 5.3. The path between* t_1 *and* t_3 *is* $\pi = \{t_1, p_1, t_2, p_2, t_3\}$*, according to the definition of gains,* $G(\pi) = \frac{Post(p_1,t_1)\cdot Post(p_2,t_2)}{Pre(p_1,t_2)\cdot Pre(p_2,t_3)} = \frac{2\cdot 1}{1\cdot 4} = 1/2$*. It means that if* t_1 *fires once,* t_3 *can fire* $1/2$ *times (in the case that* p_1 *and* p_2 *are empty initially).*

In the initial state, path π *is empty, i.e.,* $\boldsymbol{m}_0[p_1] = 0$*,* $\boldsymbol{m}_0[p_2] = 0$*. In order to reach a marking* $\boldsymbol{m}$*, such that* $\boldsymbol{m}[p_1] = 1$*,* $\boldsymbol{m}[p_2] = 1$*, so* $\boldsymbol{\sigma} = [1\ 1\ 0]^T$*,* t_1 *needs to fire once, therefore, the weighted marking of* π *under* $\boldsymbol{m}$ *is* $M(\pi, \boldsymbol{m}) = 1$*.*

Assume that from $\boldsymbol{m}$ *the intermediate transition* t_2 *fires in a maximal amount that is equal to 1, the enabling degree of* t_3 *becomes* $1/2$*, obviously it is equal to* $G(\pi) \cdot M(\pi, \boldsymbol{m})$*.*

Transition Reduction Rule (T-RR). *Let* t_j *be a transition in a continuous CF net system* $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$*, with* $|{}^\bullet t_j| = n$*,* $|t_j{}^\bullet| = k$*. Let us denote its inputs by* $P_{in} = {}^\bullet t_j$*, and its outputs by* $P_{out} = t_j{}^\bullet$*. Let* $p_x \in P_{in}$*,* $p_y \in P_{out}$*. Transition* t_j *with its input and output places can be reduced to* $n \cdot k$ *places, obtaining the reduced system* $\mathcal{S}' = \langle \mathcal{N}', \boldsymbol{m}_0' \rangle$*, by using the following process:*

(1) Replace each elementary path $\{p_x, t_j, p_y\}$ *with a place* p_{x_y}*.*

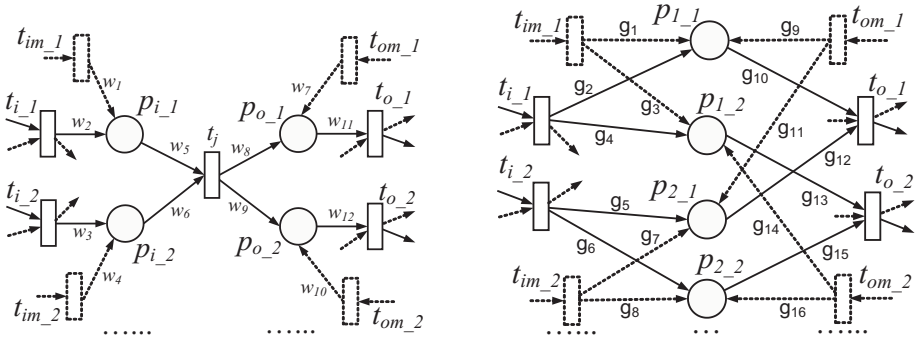
(2) Add arcs such that ${}^\bullet p_{x\_y} = {}^\bullet p_x \cup {}^\bullet p_y$*,* $p_{x\_y}{}^\bullet = p_y{}^\bullet$*.*

(3) Add weights such that $G(\pi(t_{in}, t_{out})) = G(\pi'(t_{in}, t_{out}))$*, where* $t_{in} \in {}^\bullet P_{in} \cup {}^\bullet P_{out}$*,* $t_{out} \in P_{out}{}^\bullet$*,* $\pi(t_{in}, t_{out})$ *and* $\pi'(t_{in}, t_{out})$ *are the paths from* t_{in} *to* t_{out}*, in* $\mathcal{S}$ *and* $\mathcal{S}'$ *respectively.*

(4) Put the initial marking $\boldsymbol{m}_0'[p_{x\_y}] = \boldsymbol{Post}(p_{x\_y}, t_{in}) \cdot M(\pi, \boldsymbol{m}_0)$*, where* $\pi = \{t_{in}, p_x, t_j, p_y, t_{out}\}$*.*

In step (3), the gains of paths should be maintained by putting appropriate weights on the arcs. Let us remark that the weights on the arcs can be scaled and the same behaviros are obtained. For instance, in the CPN in Fig. 5.3, to keep the gain of path $\{t_1, p_1, t_2\}$, we can put weight $\boldsymbol{Post}(p_1, t_1) = 4$ and $\boldsymbol{Pre}(p_1, t_2) = 2$ (in this case, the marking of p_1 is still zero). Obviously, the overall behaviors of the system are not changed.

Example 5.3.6. *Consider the CF net system $\mathcal{S}$ in Fig. 5.4(a), by applying T-RR to reduce t_j, the system in Fig. 5.4(b) is obtained. In the original system, transition t_j has two inputs $P_{in} = \{p_{i\_1}, p_{i\_2}\}$ and two outputs $P_{out} = \{p_{o\_1}, p_{o\_2}\}$, therefore $n = k = 2$. Transitions t_{i_1} and t_{i_2} are the inputs of p_{i_1} and p_{i_2} which may have more inputs denoted by t_{im_1} and t_{im_2}. Transitions t_{o_1} and t_{o_2} are the outputs of p_{o_1} and p_{o_2} which may also have more inputs denoted by t_{om_1} and t_{om_2}.*



(a) Original system $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ (b) Reduced system $\mathcal{S}' = \langle \mathcal{N}', \boldsymbol{m}_0{}' \rangle$

Figure 5.4: Transition reduction rule (T-RR): reducing t_j

In the reduced system $\mathcal{S}'$ there are four new places, p_{1_1}, p_{1_2}, p_{2_1} and p_{2_2}. In particular, p_{1_1} is the reduction of path $\{p_{i\_1}, t_j, p_{o\_1}\}$, p_{1_2} is the reduction of path $\{p_{i\_1}, t_j, p_{o\_2}\}$, etc. Observe that the gain of the path from t_{i_1} to t_{o_1}, i.e., $\pi = \{t_{i\_1}, p_{i\_1}, t_j, p_{o\_1}, t_{o\_1}\}$ is $G(\pi) = \frac{w_2 \cdot w_8}{w_5 \cdot w_{11}}$. The weights g_2, g_{10} on the paths of the reduced net between the same transitions, i.e., $\pi' = \{t_{i\_1}, p_{1\_1}, t_{o\_1}\}$, should satisfy $\frac{g_2}{g_{10}} = G(\pi)$. Considering p_{1_1} in $\mathcal{S}'$ for example, step (4) implies that $\boldsymbol{m}_0{}'[p_{1\_1}] = g_2 \cdot M(\pi, \boldsymbol{m}_0)$.

Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ and $\mathcal{S}' = \langle \mathcal{N}', \boldsymbol{m}_0{}' \rangle$ be the original and reduced CF net systems, σ be a firing sequence in $\mathcal{S}$. Sequence ς is said to be the projection of σ from $\mathcal{S}$ to $\mathcal{S}'$ when ς is obtained from σ by removing the elements corresponding to transitions t_j, $t_j \notin T \cap T'$.

Proposition 5.3.7. *Let $\mathcal{S}$ be a continuous CF net system, and $\mathcal{S}'$ be its reduced system obtained by applying T-RR, removing transition t_j. Assume σ is a firing sequence of $\mathcal{S}$, and ς is its projection to $\mathcal{S}'$. Then σ is fireable in $\mathcal{S}$ if and only if ς is fireable in $\mathcal{S}'$.*

Proof: Let us first consider a given firing sequence σ, and prove that σ is fireable in $\mathcal{S}$ iff ς is fireable in $\mathcal{S}'$. The proof can be easily extended to any firing sequence by using a similar argument.

Consider T-RR applied in Fig. 5.4 to reduce transition t_j and its input/output places. For the sake of simplicity, we consider a representative firing sequence in $\mathcal{S}$ composed by transitions t_{i_1}, t_{i_2}, t_j, t_{om_1} and t_{o_1}, $\sigma = t_{i\_1}(\alpha_1) t_{i\_2}(\alpha_2) t_j(\beta)\ t_{om\_1}(\alpha_3)$

$t_{o\_1}(\alpha_4)$, and its projection to the reduced system $\mathcal{S}'$ is $\varsigma = t_{i\_1}(\alpha_1)t_{i\_2}(\alpha_2)t_{om\_1}(\alpha_3)$ $t_{o\_1}(\alpha_4)$. For other transitions, since the gains of all paths between transition should be reserved according the reduction rules, consider t_{im_1} is similar to consider t_{i_1}, putting tokens to p_{i_1} with different weights; for the same reason, consider t_{im_2} is similar to consider t_{i_2}. If t_j fires tokens are put into p_{o_1} and p_{o_2} at the same time just with different weights, so consider transition t_{o_1} is similar to consider t_{o_2}.

In $\mathcal{S}$, let $\pi_1 = \{t_{i\_1}, p_{i\_1}, t_j, p_{o\_1}, t_{o\_1}\}$ and $\pi_2 = \{t_{i\_2}, p_{i\_2}, t_j, p_{o\_1}, t_{o\_1}\}$; In $\mathcal{S}'$, let π_1' and π_2' be the paths corresponding to the same transitions as π_1, π_2, respectively, i.e., $\pi_1' = \{t_{i\_1}, p_{1\_1}, t_{o\_1}\}$ and $\pi_2' = \{t_{i\_2}, p_{2\_1}, t_{o\_1}\}$.

Let us first consider a subsequence of σ, $\sigma_1 = t_{i\_1}(\alpha_1)t_{i\_2}(\alpha_2)t_j(\beta)t_{om\_1}(\alpha_3)$, and its corresponding projection to $\mathcal{S}'$, $\varsigma_1 = t_{i\_1}(\alpha_1)t_{i\_2}(\alpha_2)t_{om\_1}(\alpha_3)$. Obviously, σ_1 is fireable in $\mathcal{S}$ iff ς_1 is fireable in $\mathcal{S}'$ because transitions t_{i_1}, t_{i_2} and t_{om_1} have the same input places and corresponding markings in $\mathcal{S}$ and $\mathcal{S}'$.

In $\mathcal{S}$, if t_j fires with the maximal amount in σ_1, t_{o_1} will get the maximal enabling degree. Therefore by firing σ_1, the enabling degree of t_{o_1} can be maximally increased by:

$$\phi = \min\{\alpha_1 \cdot G(\pi_1) + \alpha_3 \cdot \frac{w_7}{w_{11}},\ \alpha_2 \cdot G(\pi_2) + \alpha_3 \cdot \frac{w_7}{w_{11}}\}$$

Considering the initial marking $\boldsymbol{m}_0$, the maximal enabling degree of t_{o_1} by firing of σ_1 is:

$$\min\{G(\pi_1) \cdot M(\pi_1, \boldsymbol{m}_0),\ G(\pi_2) \cdot M(\pi_2, \boldsymbol{m}_0)\} + \phi$$

In $\mathcal{S}'$, the enabling degree of t_{o_1} under the initial marking is equal to:

$$\min\left\{\frac{\boldsymbol{m}_0'[p_{1\_1}]}{g_{10}}, \frac{\boldsymbol{m}_0'[p_{2\_1}]}{g_{12}}\right\}$$

According to according the reduction step (4), it is equal to

$$\begin{aligned}&\min\left\{\frac{g_2 \cdot M(\pi_1, \boldsymbol{m}_0)}{g_{10}},\ \frac{g_5 \cdot M(\pi_2, \boldsymbol{m}_0)}{g_{12}}\right\}\\ &= \min\{G(\pi_1) \cdot M(\pi_1, \boldsymbol{m}_0),\ G(\pi_2) \cdot M(\pi_2, \boldsymbol{m}_0)\}\end{aligned}$$

By the firing of ς_1, it is increased by the same amount ϕ as in $\mathcal{S}$, because $G(\pi_i) = G(\pi_i')$, $i = 1, 2$ and $w_7/w_{11} = g_9/g_{10} = g_{11}/g_{12}$.

Therefore, if σ is fireable in $\mathcal{S}$, ς is for sure fireable in $\mathcal{S}'$. The other direction, if ς is fireable in $\mathcal{S}'$, σ is fireable in $\mathcal{S}$ when the intermediate transition t_j fires in the maximal amount.

A similar proof can be achieved for any firing sequence following the procedure: 1) any sequence that consists of the transitions whose input places are the same in $\mathcal{S}$ and $\mathcal{S}'$ (like t_{i_1}, t_{i_2} in Fig.5.4), is fireable in $\mathcal{S}$ iff its projection in $\mathcal{S}'$ is fireable; 2) any other transitions (like t_{o_1}, t_{o_2} in Fig.5.4) can get the same enabling degrees in $\mathcal{S}$ and $\mathcal{S}'$, when sequences in 1) fire. ■

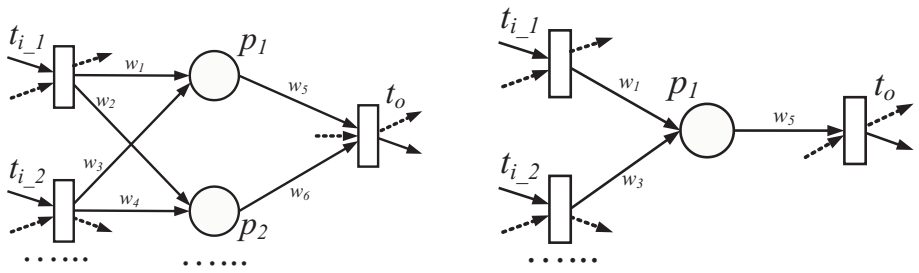
Remark 5.3.8. *It can be observed that, each time T-RR is applied to a subnet formed by paths between $T_{in} \in T$ and $T_{out} \in T$, one transition $t \notin T_{in} \cup T_{out}$ is*

removed. Therefore the repetitive application of T-RR results in a set of places between T_{in} and T_{out} but no transition.

Place Reduction Rule (P-RR). *Let p_1, p_2 be two places in a continuous CF net system, such that ${}^\bullet p_1 = {}^\bullet p_2 = T_{in} \subseteq T, p_1{}^\bullet = p_2{}^\bullet = t_{out}$. If for any $t_{in} \in T_{in}$, paths $\pi_a = \{t_{in}, p_1, t_{out}\}$ and $\pi_b = \{t_{in}, p_2, t_{out}\}$ have the same gain, i.e., $G(\pi_a) = G(\pi_b)$. Then, if $\frac{\boldsymbol{m}_0[p_1]}{\boldsymbol{Pre}(p_1,t_{out})} \leq \frac{\boldsymbol{m}_0[p_2]}{\boldsymbol{Pre}(p_2,t_{out})}$, p_2 can be removed, otherwise, p_1 can be removed.*

In order to apply P-RR, $G(\pi_a) = G(\pi_b)$ has to be satisfied. Notice that if $G(\pi_a) \neq G(\pi_b)$, it implies that the system is not live or not bounded. In particular, if $G(\pi_a) > G(\pi_b)$ then place p_1 is not bounded, otherwise the net system is not live; if $G(\pi_a) < G(\pi_b)$ then place p_2 is not bounded, otherwise the net system is not live either.

Example 5.3.9. *Fig. 5.5(a) shows a CF net system in which $T_{in} = \{t_{i\_1}, t_{i\_2}\}$. In order to apply P-RR, the weights of arcs should satisfy $\frac{w_1}{w_5} = \frac{w_2}{w_6}$, and $\frac{w_3}{w_5} = \frac{w_4}{w_6}$. Assume $\frac{\boldsymbol{m}_0[p_1]}{w_5} \leq \frac{\boldsymbol{m}_0[p_2]}{w_6}$, then by removing p_2, the reduced system is shown in Fig. 5.5(b).*



(a) Original subsystem $\mathcal{S}$ (b) Reduced subsystem $\mathcal{S}'$

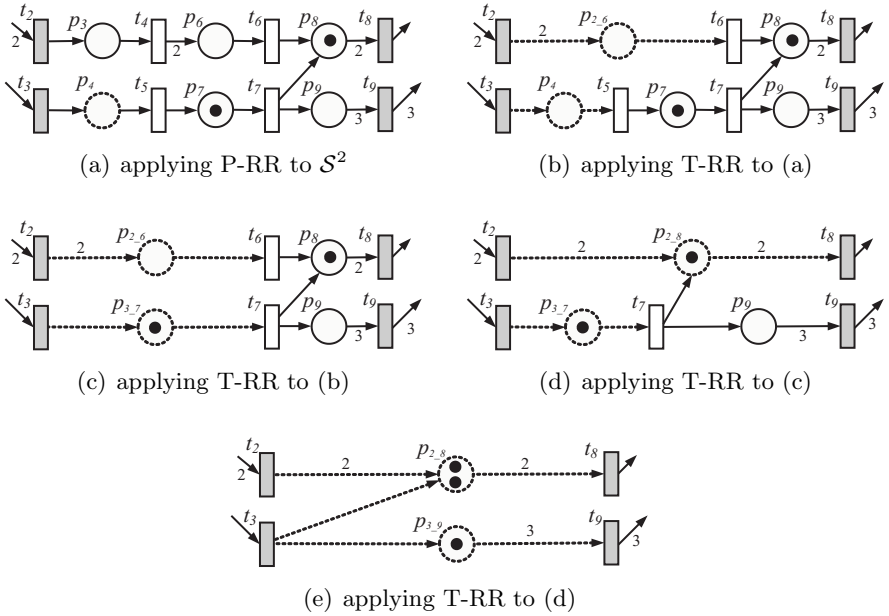
Figure 5.5: Place Reduction Rule (P-RR): reducing p_2

Proposition 5.3.10. *Let $\mathcal{S}$ be a continuous CF net system, and $\mathcal{S}'$ be the reduced system obtained by applying P-RR, sequence σ is fireable in $\mathcal{S}$ if and only if σ is fireable in $\mathcal{S}'$.*

Proof: It is easy to verify that the places being removed by applying P-RR belong to a particular type of *implicit places*, i.e., those places that never uniquely restrict the firing of its output transitions (see [90]). Therefore, they can be removed without affecting the behavior of the rest of the system. ■

Example 5.3.11. *Let us apply the reduction rules on subsystem $\mathcal{S}^2$ in Fig. 5.2(a). The net system in Fig. 5.6(a) is obtained by applying P-RR to remove place p_5. By applying T-RR to the path between t_2 and t_6, p_{2_6} is obtained (Fig.5.6(b)). Similarly, the application of T-RR to the path between t_3 and t_7 in Fig.5.6(b), removes t_5 and*

obtains p_{3_7} (Fig.5.6(c)). The application of T-RR to the path between t_2 and t_8 in Fig.5.6(c), removes t_6 and obtains p_{2_8} (Fig.5.6(d)). The application of T-RR to the path between t_3 and t_9 in Fig.5.6(d), removes t_7 and obtains p_{3_9} (Fig.5.6(e)). Finally, only two places are left with markings $\boldsymbol{m}_0'[p_{2\_8}] = 2$, $\boldsymbol{m}_0'[p_{3\_9}] = 1$. The reduced subsystem in Fig.5.6(e) is the abstraction of $\mathcal{S}^2$.



(a) applying P-RR to $\mathcal{S}^2$

(b) applying T-RR to (a)

(c) applying T-RR to (b)

(d) applying T-RR to (c)

(e) applying T-RR to (d)

Figure 5.6: Reduction process of $\mathcal{S}^2$ in Fig. 5.2(a)

Let us point out that if we apply the classical reduction rules proposed for discrete nets (see, for example, [13]) to the example shown in Ex. 5.3.11, the same reduced net system can be obtained. However, in this work we extend the classical reduction rules to CPNs where the markings, also the weights, are real numbers. Let us consider the following simple example:

Example 5.3.12. *Given the a (partial) CPN system shown in Fig. 5.7(a) and it is assumed that we want to reduce the paths between transition t_a and t_b. By applying T-RR to remove transition t_1, Fig.5.7(b) is obtained; similarly, t_2 can also be removed, obtaining Fig.5.7(c) which is clearly equivalent to Fig.5.7(d).*

Let us point out that the main limitation of applying the classical reduction rules to this example is that, in those rules for discrete nets fractional firings are not considered. For example, after the reduction, the marking of the single place left between t_a and t_b and the weights on arcs are decimal fractions, which are forbidden in discrete cases.

Assume that, using T-RR and P-RR, we reduce the paths between two sets of transitions T_{in} and T_{out}. Now we will discuss the uniqueness of the fully reduced system.

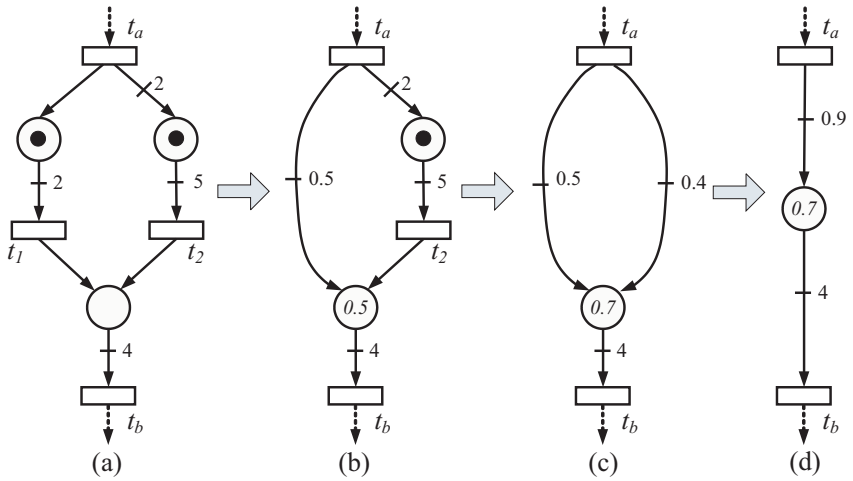


Figure 5.7: Reduction of the paths between t_a and t_b using T-RR: the classical reduction rules for discrete nets cannot be applied

Property 5.3.13. *Any arbitrary and interleaved application of T-RR and P-RR until none of them can be applied produces the same reduced system.*

Proof: It is first proved that the order of adjacent rules that are applied can be interchanged, obtaining the same reduced system. Otherwise stated, let A and B be the instances of two rules, by applying AB or BA, the same system is obtained. Then we will show that any sequence of rules, leading to the fully reduced system, can be reordered. After that, the uniqueness of the reduced system can be easily proved.

1) if A and B are both instances of T-RR (or P-RR), it is trivial.

2) if A and B are instances of different rules. Without loss of generality, assume A is an instance of T-RR, removing a transition t_j and B is an instance of P-RR, removing an implicit place p_x. Obviously, if $t_j \notin {}^\bullet p_x \cup p_x{}^\bullet$, A and B are independent, so the system obtained after applying AB is equivalent to the one obtained after applying BA. Therefore, we only need to consider the two cases shown in Fig.5.8, where t_j can be removed by using T-RR, at the same time, its input or/and output places can be reduced by using P-RR. Its extension to more general structures is quite straightforward.

We will show that for case (a), by applying AB and BA, the same system is obtained. The analysis to case (b) is similar.

Since p_x can be removed by using P-RR, then $w_1/w_3 = w_2/w_4$ and in the initial state $\boldsymbol{m}_0[p_x]/w_3 \geq \boldsymbol{m}_0[p_1]/w_4$. Let path $\pi_1 = \{t_1, p_x, t_j, p_2, t_2\}$ and $\pi_2 = \{t_1, p_1, t_j, p_2, t_2\}$, then we have the weighted marking $M(\pi_1, \boldsymbol{m}_0) \geq M(\pi_2, \boldsymbol{m}_0)$.

If first T-RR has been applied to remove t_j, the system in Fig.5.9(a) is obtained. Let us first consider the obtained place p_a and p'_a. Without loss of generality, we should have: $\frac{g_1}{g_5} = \frac{w_1 \cdot w_5}{w_3 \cdot w_7} = \frac{g_2}{g_6} = \frac{w_2 \cdot w_5}{w_4 \cdot w_7}$, moreover, with the initial marking $\boldsymbol{m}_0{}'[p'_a] =$

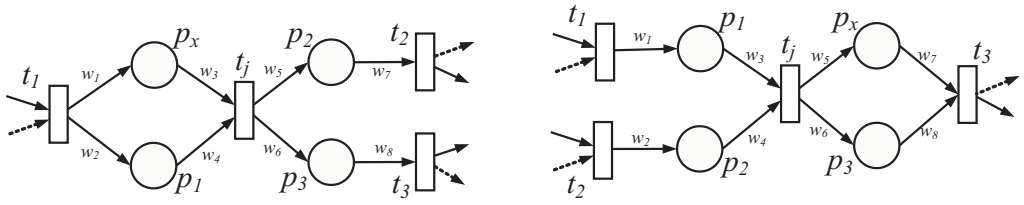


Figure 5.8: The two cases with $t_j \in {}^\bullet p_x \cup p_x{}^\bullet$

$g_1 \cdot M(\pi_1, \boldsymbol{m}_0)$ and $\boldsymbol{m}_0{}'[p_a] = g_2 \cdot M(\pi_2, \boldsymbol{m}_0)$, therefore, $\frac{\boldsymbol{m}_0{}'[p'_a]}{g_5} \geq \frac{\boldsymbol{m}_0{}'[p_a]}{g_6}$, p'_a is implicit place. Then, it can be removed by applying P-RR. Similarly, for p_b and p'_b, let $\frac{g_3}{g_7} = \frac{w_2 \cdot w_6}{w_4 \cdot w_8}$, $\frac{g_4}{g_8} = \frac{w_1 \cdot w_6}{w_3 \cdot w_8}$, p'_b is also implicit and can be removed. The obtained system is shown in Fig.5.9(c).

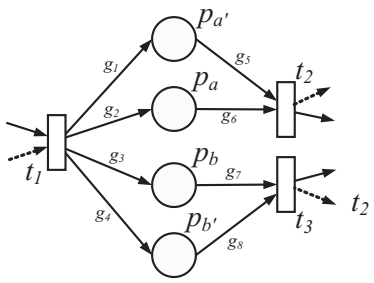
If first P-RR has been applied to remove p_x, the system in Fig.5.9(b) is obtained. Then by applying T-RR, t_j is removed, it is clear that the same reduced system in Fig.5.9(c) is achieved.

Now we know that the order of applying reduction rules is not important. Let Γ_1 and Γ_2 be two sequences of rules leading to two fully reduced systems $\mathcal{S}^1$ and $\mathcal{S}^2$. It is clear that, the same number of T-RR is applied in Γ_1 and Γ_2 (because applying T-RR once, one transition between T_{in} and T_{out} is removed). From 1) and 2), we can transform the sequence Γ_1 to Γ'_1 by interchanging the order of adjacent rules, until all the instances of T-RR are moved ahead of instances of P-RR. Assume that by applying all the instances of T-RR, the obtained system is $\mathcal{S}'_1$. On the other hand, we can also transform the sequence Γ_2 to Γ'_2 by doing the same interchanging and assume that by applying all the instances of T-RR, the obtained system is $\mathcal{S}'_2$. Obviously, $\mathcal{S}'_1$ and $\mathcal{S}'_2$ are equivalent, and there are only places (but no transition) left between T_{in} and T_{out}. After that, the instances of P-RR are applied to reduce implicit places in $\mathcal{S}'_1$ and $\mathcal{S}'_2$. If they are fully reduced, for sure the finally obtained systems are the same, i.e., $\mathcal{S}_1$ and $\mathcal{S}_2$ are equivalent. Therefore, the fully reduced system is unique. ■

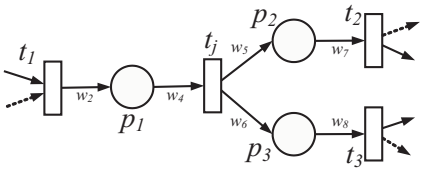
Remark 5.3.14. *In order to obtain the fully reduced system, we need to explore the paths between transitions. Concerning the computational complexity, it is suggested that before considering to apply T-RR, we should first apply P-RR as much as possible to remove the implicit places. For example, in Ex.5.3.11, P-RR is first applied to remove the implicit place* p_5.

5.3.3 Complemented subsystems

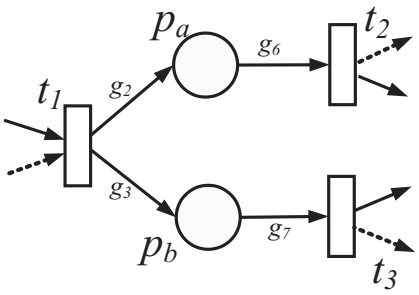
Definition 5.3.15. *Let* $\mathcal{S}$ *be a continuous CF net system, and* $\mathcal{S}^i$, $i = 1, 2$ *be the subsystems obtained by cutting through places* $B \in P$. *The complemented subsystems, denoted by* $\mathcal{CS}^i$, *is obtained from* $\mathcal{S}$ *by substituting* $\mathcal{S}^j$, $j = 1, 2$, $j \neq i$ *with its abstraction.*



(a) First apply T-RR



(b) First apply P-RR



(c) Fully reduced

Figure 5.9: Reduction by applying rules in different order

Let us still consider the system in Fig.5.2(a). By applying the proposed rules, the paths in $\mathcal{S}^1$ between interface transitions t_1 and t_{10} can be reduced to a single place p_{10_1}, obtaining the abstraction of $\mathcal{S}^1$. Using this abstraction to complement $\mathcal{S}^2$, the complemented subsystem $\mathcal{CS}^2$ is obtained, shown in Fig.5.2(c). Similarly, the abstraction of $\mathcal{S}^2$ can be constructed, and the complemented subsystem $\mathcal{CS}^1$ is shown in Fig.5.2(b). Notice that, the cutting places and interface transitions are shared in both complemented subsystems.

Remark 5.3.16. *A direct consequence of Proposition. 5.3.7 and 5.3.10 is that the firing sequences and reachable markings of the original system are preserved in its complemented subsystems.*

The decomposition method can be easily extended to a large scale system that is decomposed into K subsystems, by given sets of cutting places. The complemented subsystems are constructed in two steps: first, each subsystem builds its abstraction (the reduced subsystem respect to its interface transitions); then, each subsystem constructs its complementing parts based on the abstractions of the rest of the system that have been built in the first step. In this way, each subsystem does not need the detailed structures and states of other parts of the system, but only their abstractions.

5.3.4 The control law computation

A interesting property of the complemented subsystem is that their firing sequences are identical to those of the original system (Remark 5.3.16). Thus, the local control laws can be computed separately, driving all subsystems to their corresponding final states. However, local control laws may not be compatible with each other, i.e., there may exist a interface transition that does not fire with the same amount in the corresponding complemented subsystems (see Ex.5.3.17 for a example). In order to overcome this problem, a coordinator is introduced (see the control structure in Fig. 5.1). Local controllers will send limited information (the local control law and the minimal T-*semiflow*) to the coordinator. Algorithms are proposed to compute a globally admissible control law based on this information, without knowing the detailed structures of subsystems but only the interface transitions.

Example 5.3.17. *Let us consider the CF net in Ex. 5.3.2 and the two obtained complemented subsystems in Fig.5.2(b) and Fig.5.2(c). The initial and final marking* $\boldsymbol{m}_0$, $\boldsymbol{m}_f$ *of the original system, and its corresponding minimal firing count vector* $\boldsymbol{\sigma}_{min}$ *are shown in Table 5.1. As for the subsystems, the minimal firing count vectors* $\boldsymbol{\sigma}^i_{min}$ *of* $\mathcal{CS}^i$ *for reaching the corresponding final marking* $\boldsymbol{m}^i_f$ *from* $\boldsymbol{m}^i_0$ *are computed separately, they are also given in Table 5.1. It can be observed that* $\boldsymbol{\sigma}^1_{min}$ *and* $\boldsymbol{\sigma}^2_{min}$ *are not compatible, because their interface transitions do not have the same firing counts, for instance,* $\boldsymbol{\sigma}^1_{min}[t_1] \neq \boldsymbol{\sigma}^2_{min}[t_1]$.

Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be the original system, with $\boldsymbol{m}_f > 0$ the desired final state. It is assumed that $\mathcal{S}$ is decomposed into K subsystems, $\mathcal{S}^1$ to $\mathcal{S}^k$. The following notations are used:

(1) $\boldsymbol{\sigma}_{min}$: the minimal firing count vector driving $\mathcal{S}$ to $\boldsymbol{m}_f$.

(2) $B^{(i_1,i_2)}$: the buffer cutting places between $\mathcal{S}^{i_1}$ and $\mathcal{S}^{i_2}$.

(3) $U^{(i_1,i_2)}$: the interface transitions between $\mathcal{S}^{i_1}$ and $\mathcal{S}^{i_2}$.

(4) $\mathcal{CS}^i = \langle \mathcal{CN}^i, \boldsymbol{m}_0{}^i \rangle$: the complemented subsystems with corresponding final state $\boldsymbol{m}^i_f$, $i = 1, 2, ..., K$.

(5) $\boldsymbol{x}^i$: the minimal T-*semiflow* in $\mathcal{CN}^i$, $i = 1, 2, ..., K$.

(6) $\boldsymbol{\sigma}^i_{min}$: the minimal firing count vector driving $\mathcal{CS}^i$ to $\boldsymbol{m}^i_f$, $i = 1, 2, ..., K$.

According to the decomposition and reduction process, the obtained complemented CF subnets are also consistent and conservative. Therefore, the minimal T-*semiflow* and minimal firing count vector are unique [93, 104], i.e., $\boldsymbol{x}^i$ and $\boldsymbol{\sigma}^i_{min}$ are unique. So, any firing count vector $\boldsymbol{\sigma}^i$ driving $\mathcal{CS}^i$ to its final state can be written as follows

$$\boldsymbol{\sigma}^i = \boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i, \quad \alpha^i \geq 0 \tag{5.1}$$

Table 5.1: Markings and firing count vectors of the systems in Fig. 5.2

P	$\boldsymbol{m}_0(\boldsymbol{m}_f)$	${\boldsymbol{m}_0}^1({\boldsymbol{m}_f}^1)$	${\boldsymbol{m}_0}^2({\boldsymbol{m}_f}^2)$	T	$\boldsymbol{\sigma}_{min}$	$\boldsymbol{\sigma}^1_{min}$	$\boldsymbol{\sigma}^2_{min}$
p_1	0 (0.4)	0 (0.4)	0 (0.4)	t_1	1.4	0.9	1.4
p_2	0 (0.3)	0 (0.3)	0 (0.3)	t_2	0.55	0.3	0.55
p_3	0 (0.3)		0 (0.3)	t_3	1	0.5	1
p_4	0 (0.3)		0 (0.3)	t_4	0.25		0.25
p_5	1 (1.3)		1 (1.3)	t_5	0.7		0.7
p_6	0 (0.5)		0 (0.5)	t_6	0		0
p_7	1 (0.3)		1 (0.3)	t_7	1.4		1.4
p_8	1 (0.4)		1 (0.4)	t_8	1	0.5	1
p_9	0 (0.2)		0 (0.2)	t_9	0.4	0.23	0.4
p_{10}	0 (0.6)	0 (0.6)	0 (0.6)	t_{10}	0.8	0.3	0.8
p_{11}	0 (0.2)	0 (0.2)		t_{11}	0.6	0.1	
p_{12}	0 (0.1)	0 (0.1)		t_{12}	0.7	0.2	
p_{13}	0 (0.1)	0 (0.1)		t_{13}	0.35	0.1	
p_{14}	0 (0.3)	0 (0.3)		t_{14}	0.5	0	
p_{15}	0 (0.1)	0 (0.1)		t_{15}	0.6	0.1	
p_{16}	0 (0.1)	0 (0.1)		t_{16}	0.25	0	
p_{17}	1 (0.1)	1 (0.1)					
p_{18}	1 (0.2)	1 (0.2)					
p_{19}	1 (0.1)	1 (0.1)					
p_{2_8}		2 (2.1)					
p_{3_9}		1 (0.8)					
p_{10_1}			1 (0.4)				

Algorithm 6 is used by the coordinator controller. Non-negative value α^1, α^2, ..., α^K are obtained by solving a simple LPP. Then these values are sent back to local controllers. It is ensured that by updating the local control law from $\boldsymbol{\sigma}^i_{min}$ to $\boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i$, the interface transitions fire in the same amounts in corresponding neghboring subsystems.

Given a reachable final state $\boldsymbol{m}_f$, LPP (5.2) is feasible. Let $\boldsymbol{\sigma}$ be a firing count vector driving $\mathcal{S}$ to $\boldsymbol{m}_f$, and denote by $\boldsymbol{\sigma}^{i_1}$ and $\boldsymbol{\sigma}^{i_2}$ the projections of $\boldsymbol{\sigma}$, corresponding to $\mathcal{CS}^{i_1}$ and $\mathcal{CS}^{i_2}$. By firing $\boldsymbol{\sigma}^{i_1}$ and $\boldsymbol{\sigma}^{i_2}$ in $\mathcal{CS}^{i_1}$ and $\mathcal{CS}^{i_2}$, markings ${\boldsymbol{m}_f}^{i_1}$, ${\boldsymbol{m}_f}^{i_2}$ are reached. Obviously, the transitions in $U^{(i_1,i_2)}$ fire in the same amounts in $\boldsymbol{\sigma}^{i_1}$ and $\boldsymbol{\sigma}^{i_2}$, so there exist α^{i_1} and α^{i_2}, satisfying the constraints of LPP (5.2).

Proposition 5.3.18. *Let α^i be the value obtained by using Algorithm 6 and $\boldsymbol{\sigma}^i = \boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i$, $i = 1, 2, ..., K$ be the local control laws of $\mathcal{CS}^i$. The global control law $\boldsymbol{\sigma}$ obtained by merging all the local ones, is the (unique) minimal firing count vector driving $\mathcal{S}$ to $\boldsymbol{m}_f$.*

Proof: It is trivial that $\boldsymbol{\sigma}$ can drive $\mathcal{S}$ to $\boldsymbol{m}_f$. If $\boldsymbol{\sigma}$ is not the minimal one, some amounts of T-*semiflow* can be subtracted, obtaining a contradiction with the

Algorithm 6 Coordinator

Input: $\boldsymbol{\sigma}^i_{min}$, $\boldsymbol{x}^i$, $i = 1, 2, .., K$
Output: α^i, $i = 1, 2, .., K$

1: Receive $\boldsymbol{\sigma}^i_{min}$ and $\boldsymbol{x}^i$ from local controllers
2: Compute α^i by solving LPP:

$$\begin{aligned} \min \quad & \sum_{i=1}^{K} \alpha^i \\ \text{s.t.} \quad & \boldsymbol{\sigma}^{i_1}_{min}[t_j] + \alpha^{i_1} \cdot \boldsymbol{x}^{i_1}[t_j] = \boldsymbol{\sigma}^{i_2}_{min}[t_j] + \alpha^{i_2} \cdot \boldsymbol{x}^{i_2}[t_j], \forall t_j \in U^{(i_1,i_2)} \\ & \forall i_1, i_2 \in \{1, 2, ..., K\}, \mathcal{CS}^{i_1} \text{ and } \mathcal{CS}^{i_2} \text{ are neighbors.} \\ & \alpha^i \geq 0, i = 1, 2, ..., K \end{aligned} \tag{5.2}$$

3: Send α^i to $\mathcal{CS}^i$;

objective function of LPP (5.2). ■

Notice that the minimal firing count vector is unique, implying that the solution of LPP (5.2) is also unique.

Algorithm 7 is used by the local controllers. In the first step, the minimal firing count vector $\boldsymbol{\sigma}^i_{min}$ of each subsystem $\mathcal{CS}^i$ is computed separately by the local controller. Then, every subsystem $\mathcal{CS}^i$ sends $\boldsymbol{\sigma}^i_{min}$ to the coordinator, together with its corresponding minimal T-*semiflow* (only once if the net structure does not change). After α^i is received from the coordinator, the controller of $\mathcal{CS}^i$ can be implemented independently by considering $\boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i$. In particular, the minimum-time ON-OFF controller (presented in Chapter 3) is used.

Algorithm 7 Local Controller i

Input: $\mathcal{CN}^i$, $\boldsymbol{m}^i_0$, $\boldsymbol{m}^i_f$
Output: $\boldsymbol{\sigma}^i$

1: Compute $\boldsymbol{\sigma}^i_{min}$ the drives the system to $\boldsymbol{m}^i_f$;
2: Compute the minimal T-*semiflow* $\boldsymbol{x}^i$;
3: Send $\boldsymbol{\sigma}^i_{min}$ and $\boldsymbol{x}^i$ to the coordinator;
4: Receive α^i from the coordinator;
5: Update $\boldsymbol{\sigma}^i \leftarrow \boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i$;
6: Apply the ON-OFF controller;

Since only limited information (the local control laws and the minimal T-semiflows) are required by the coordinator, very low communication costs are obtained (two vectors $\boldsymbol{\sigma}^i \in \mathbb{R}^{|T^i|}$ and $\boldsymbol{x}^i \in \mathbb{R}^{|T^i|}$ for each subsystem S^i). When the agreement is obtained, all the subsystems work independently.

5.3.5 A case study

In order to illustrate the developed approach, let us consider the CF net in Fig.5.10. It is adapted from the model of a simple manufacturing line that makes tables [81]. It consists of three work stations: WS_1 and WS_2 and WS_3. Two types of raw materials A and B are processed by WS_1 and WS_2 respectively. The obtained semi-products are deposited in buffers and will be finally assembled in WS_3 to make the final products. Table 5.2 gives the interpretations of the model. We will apply to this system the proposed decentralized control method.

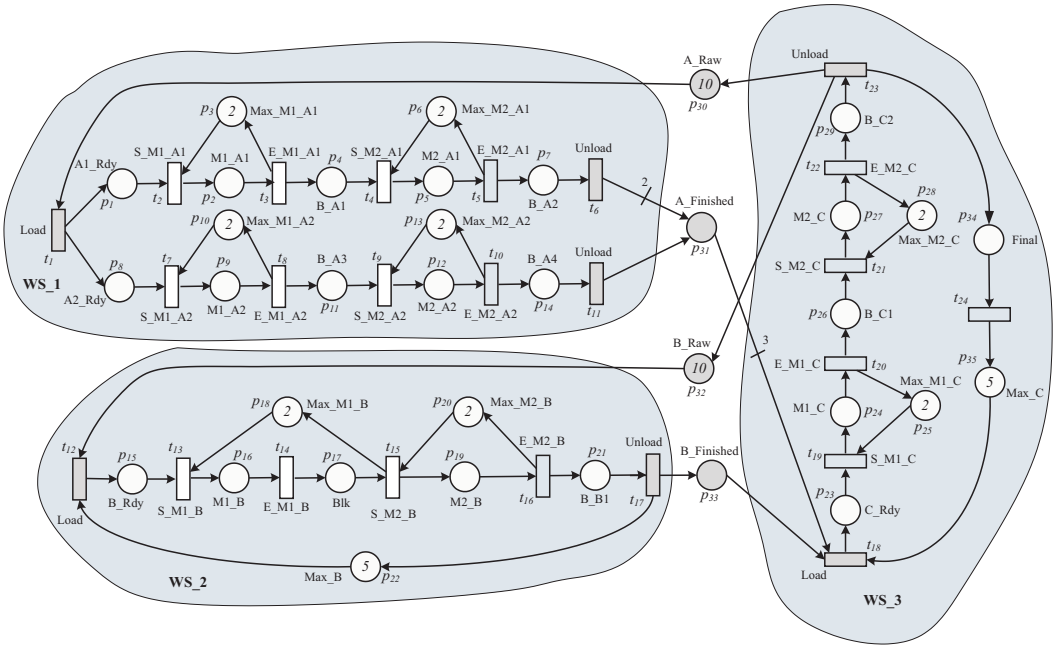


Figure 5.10: The TCPN model of a manufacturing system with three work stations.

The original system is cut into three subsystems $\mathcal{CS}^1$ to $\mathcal{CS}^3$ corresponding to work stations WS_1 to WS_3. The buffer places are $B^{(1,3)} = \{p_{30}, p_{31}\}$, $B^{(2,3)} = \{p_{32}, p_{33}\}$ and the interface transition are $U^{(1,3)} = \{t_1, t_6, t_{11}, t_{18}, t_{23},\}$, $U^{(2,3)} = \{t_{12}, t_{17}, t_{18}, t_{23}\}$. It is assumed that in the initial state both types of materials have quantities equal to 10, while two machines are available for any processing, production lines in WS_2 and WS_3 have maximal capabilities equal to 5. The firing rates are: $\lambda_8 = \lambda_{10} = 1/2$, $\lambda_{15} = \lambda_{16} = 1/3$, $\lambda_{20} = \lambda_{22} = 1/4$ and for other transitions, all equal to 1. Under this setting, the maximal throughput of transition E_M2_C (t_{22}, which models the machine that produces the final product) in the steady state is 0.33 ([87]).

The complemented subsystems are shown in Fig. 5.11, the final states of subsystems and their corresponding minimal firing count vectors are shown in Table 5.3.

In this specific example, the minimal T-semiflows of subsystems are unit vectors

Table 5.2: The interpretation of the PN model in Fig.5.10

Labels	**Interpretation**
x_Rdy	material x is ready
Mx_y	machine x processing y
Max_Mx_y	the free machine x processing y
Blk	blocked
B_x	the buffer of semi-product x
Max_x	the maximal allowed capacity of x
Final	the final product
x_Raw	raw material x
x_finish	the semi-product x finished
S_Mx_y	machine x starts to process y
E_Mx_y	machine x finishes the process of y

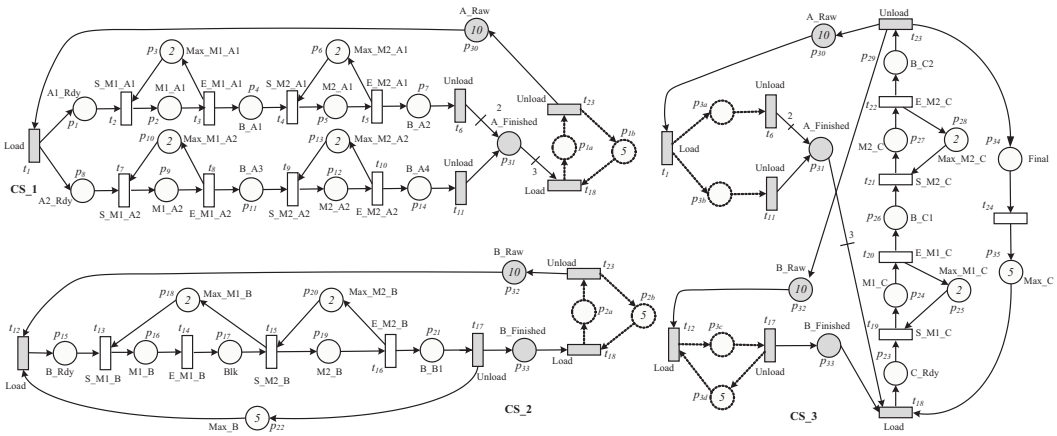


Figure 5.11: The complemented subsystems obtained from the model in Fig.5.10

1. By applying Algorithm 6, the solution is quite straightforward: $\alpha^1 = \alpha^3 = 0$ and $\alpha^2 = 0.33$. So the control law of $\mathcal{CS}^2$ should be updated to $\boldsymbol{\sigma}^2_{min} + 0.33 \cdot \mathbf{1}$, the control laws of $\mathcal{CS}^1$ and $\mathcal{CS}^3$ are $\boldsymbol{\sigma}^1_{min}$ and $\boldsymbol{\sigma}^3_{min}$, respectively. By applying the ON-OFF controller to each subsystem, the final state is reached in 17.66 time units, which is the minimum-time.

5.4 Conclusions

This chapter focuses on the minimum-time decentralized control of CF continuous Petri nets. The addressed problem is to drive the system from an initial state to a desired final one.

We assume that the original system can be divided through a given sets of places. It should be noticed that the number of interface transitions varies, depending on

Table 5.3: Final states and minimal firing count vectors of the system in Fig. 5.11

$\mathcal{CS}^1$ (WS_1)				$\mathcal{CS}^2$(WS_2)				$\mathcal{CS}^3$(WS_3)			
p	$\boldsymbol{m}_f{}^1$	t	$\boldsymbol{\sigma}^1_{min}$	p	$\boldsymbol{m}_f{}^1$	t	$\boldsymbol{\sigma}^1_{min}$	p	$\boldsymbol{m}_f{}^1$	t	$\boldsymbol{\sigma}^1_{min}$
p_1	0.33	t_1	8.17	p_{15}	0.33	t_{12}	7.00	p_{23}	0.33	t_1	8.17
p_2	0.33	t_2	7.83	p_{16}	0.33	t_{13}	6.67	p_{24}	1.33	t_6	6.50
p_3	1.67	t_3	7.50	p_{17}	1.00	t_{14}	6.34	p_{25}	0.67	t_{11}	0.00
p_4	0.33	t_4	7.17	p_{18}	0.67	t_{15}	5.34	p_{26}	0.33	t_{12}	7.33
p_5	0.33	t_5	6.83	p_{19}	1.00	t_{16}	4.34	p_{27}	1.33	t_{17}	4.33
p_6	1.67	t_6	6.50	p_{20}	1.00	t_{17}	4.00	p_{28}	0.67	t_{18}	4.00
p_7	0.33	t_7	2.00	p_{21}	0.33	t_{18}	3.67	p_{29}	0.33	t_{19}	3.67
p_8	6.17	t_8	1.33	p_{22}	2.00	t_{23}	0.00	p_{30}	2.17	t_{20}	2.33
p_9	0.67	t_9	1.00	p_{32}	3.00			p_{31}	1.00	t_{21}	2.00
p_{10}	1.33	t_{10}	0.33	p_{33}	0.33			p_{32}	3.00	t_{22}	0.67
p_{11}	0.33	t_{11}	0.00	p_{2a}	3.67			p_{33}	0.33	t_{23}	0.33
p_{12}	0.67	t_{18}	4.00	p_{2b}	1.33			p_{34}	0.33	t_{24}	0.00
p_{13}	1.33	t_{23}	0.33					p_{35}	1.00		
p_{14}	0.33							p_{3a}	1.67		
p_{30}	2.17							p_{3b}	8.17		
p_{31}	1.00							p_{3c}	3.00		
p_{1a}	3.67							p_{3d}	2.00		
p_{1b}	1.33										

how those cutting places are chosen. This may further influence the computational complexity, because the size of complemented subsystems is larger if we use a cut that introduces many interface transitions. Two rules are proposed to reduce subsystems, more specifically, the paths between interface transitions can be reduced to some places (that are implicit in the global system). In the worst case, the number of places may not be reduced, but since all intermediate transitions in the paths are removed, the subsystems are still highly simplified in general, obtaining their abstractions. A coordinator is introduced to reach the agreement among the control laws of neighboring subsystems, by solving a simple LPP. The coordinator does not need to know the detailed structures of subsystems: only limited information (the minimal firing count vector and minimal T-*semiflow*) are exchanged, ensuring a low communication cost. By applying an ON-OFF strategy in each subsystem, the global final state is reached in minimum-time.

This method has limitations when we consider a general net system. First, general reduction rules used for obtaining complemented subsystems, are not available; second, since we cannot uniquely determine the minimal firing count vector, a globally admissible control law may not be achieved if an "incorrect" one has been chosen. In the next chapter, we will propose a method for distributed control of general net systems based on the distributed MPC framework.

Chapter 6

Distributed MPC Control of General nets

In Chapter 5 we have presented a decentralized control method for CF nets. In this chapter we still focus on the target marking control problem of TCPNs in distributed setting, but now for general net systems. However, here the methods are not designed for minimum-time control. We propose a distributed control approach for general TCPNs, based on Model Predictive Control (MPC), in which an objective function is considered at each time step. Another important characteristic of the control method proposed in this chapter (also a main difference from the previous decentralized controller for CF nets) is that, no high level coordinator is needed and only few communication occurs among neighboring subsystems. We first propose a centralized MPC controller, in which asymptotic stability is guaranteed; the state trajectory is forced to be inside an interior convex subset defined by the current state ($\boldsymbol{m}_k$) and the final one ($\boldsymbol{m}_f$). Then, we apply this controller to a distributed setting, and for this aim we use a particular strategy to maintain the strict positiveness of the markings of buffer places. We prove that, by using the proposed algorithm, the desired final states of all subsystems can be reached in finite time (even if at different time instants).

6.1 Introduction

The existing distributed control methods for the target marking control problem of TCPNs are only applicable for limited subclasses of nets: the method proposed in [4] assumes that the global system as well as subsystems are mono-T-semiflow; the decentralized controller we present in Section 5 may only be applied for CF nets. The basic idea of these two approaches is similar: first, local control laws are computed independently; then, an iterative algorithm or a high level coordinator is used to achieve an agreement among subsystems by adding some T-semiflows. In this chapter, a distributed control method, based on Model Predictive Control (MPC), is presented for the target marking control of general nets. Similarly to the previous methods, we still assume a (large scale) system modelled by TCPNs that is decomposed into subsystems by sets of buffer places that facilitate the interactions among neighbors.

We first propose a centralized MPC controller, assuming the nets to be consistent. A key problem of the MPC based approaches is the stability. One classical method for achieving the close-loop stability involves in adding terminal constraints and terminal weight [70]; and another recently proposed scheme is called stability-constrained MPC, in which a stability constraint, computed at each time step, is imposed on the first state in the prediction [22]. Closely related to the second approach, in the proposed method we constrain the states to be inside a closed convex subset of the reachability space. We prove that by using this constraint asymptotic stability can be ensured. Let us remark that in the MPC controller proposed in [64] for TCPNs, the states are constrained to be on the straight line from the initial state to the final one, which in fact is a particular case of our method. Similarly, the control strategy proposed here is less constrained than the method proposed in [5], in which a linear trajectory was first considered.

Distributed control becomes particularly attractive when the system is geographically or functionally distributed, which is most frequently in the case of large scale plants. A lot of works related to distributed MPC (DMPC) can be found in the literature (see, for example, [85, 18, 91, 24]), in which subsystems are controlled in a local basis. It is usually assumed that the local controller is able to access all the variables of the corresponding subsystem and limited information of its neighbors.

Different from the method proposed in Chapter 5, the high level central coordinator is no longer needed. The topology of the communication is *partially connected*: two local controllers are able to communicate with each other if their corresponding subsystems are neighbors; thus, communication among local controllers only occurs in neighborhood, obtaining low communication costs. For instance, the sketch of a distributed control structure composed of 6 subsystems can be described as in Fig. 6.1. On the other hand, we should recall that, the method proposed here is not designed for minimum-time control; instead of that, an objective function (a quadratic function, which is mostly used in MPC approaches) is considered at each time step.

Similarly to the previous decentralized method for CF nets, the structure of a

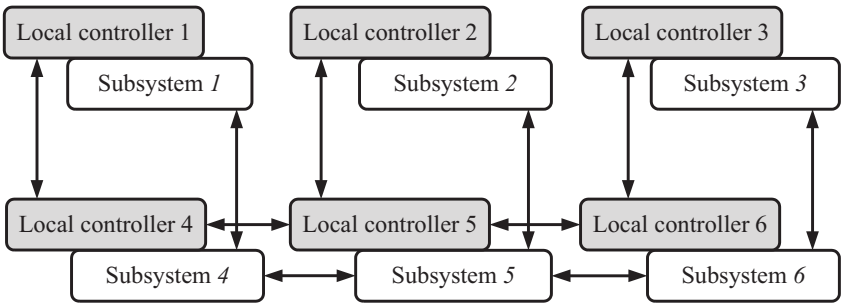


Figure 6.1: The sketch of a distributed control structure composed of 6 subsystems

subsystem can only be accessed by its corresponding local controller; at the same time, intersections among subsystems—those buffer places—can be accessed by all the neighboring subsystems that are connected to them.

One important issue of our control method is related to the buffer places: because local subsystems work independently, their markings may keep decreasing and converge to zero (for timed model, it takes infinite time); and subsystems may "stop" in certain states waiting for more tokens of the buffer places. To overcome this problem, an alternative optimization problem with a different cost function is solved in an interleaved way, putting more tokens to the buffer places with their markings converging to zero because these places may be restricting the evolution of subsystems to their final states.

6.2 A centralized MPC controller

In this section, we will present a MPC controller for TCPNs where the the states are forced to be inside a convex set, thus we prove that the convergence to the final state is guaranteed. The controller is developed based on the discrete time CPN model, represented in (4.3) with a small enough sampling period Θ satisfying (4.4) [64]. In the previous methods proposed for general net systems (non-CF nets), we assume $\boldsymbol{m}_0 > 0$. However, in the method proposed here, we conjecture that if the net is consistent and there exists no initially empty siphon, $\boldsymbol{m}_0$ is not necessarily an interior point. However, we will keep this assumption for the simplicity of presentation, and also for the convenience of comparisons with other methods. We assume that the final state ($\boldsymbol{m}_f$) is an interior point and the net is consistent.

The method proposed here is based on the MPC scheme, in which the stability is one of the main issues. The idea proposed here is that at any time instant k, we force all the predictive states to be inside a convex subset that is defined by current state $\boldsymbol{m}_k$ and the final state $\boldsymbol{m}_f$, denoted by $R(N, \boldsymbol{m}_k, \boldsymbol{m}_f)$, by means of adding the following constrains to each predictive state:

$$\begin{cases} \boldsymbol{m}_f[p_i] \geq \boldsymbol{m}_{k+j+1}[p_i] \geq \boldsymbol{m}_{k+j}[p_i], \text{ if } \boldsymbol{m}_f[p_i] \geq \boldsymbol{m}_0[p_i], j = 0, ..., N-1 \\ \boldsymbol{m}_f[p_i] \leq \boldsymbol{m}_{k+j+1}[p_i] \leq \boldsymbol{m}_{k+j}[p_i], \text{ if } \boldsymbol{m}_f[p_i] \leq \boldsymbol{m}_0[p_i], j = 0, ..., N-1 \end{cases} \tag{6.1}$$

Therefore, after each time step, the marking of every place should only move *closer* towards its final one or remain in the same marking (given by constraints (6.1)). The method proposed here is a generalization of the MPC controller introduced in [64], where linear state trajectories are considered. Comparing with the heuristic minimum-time method proposed in [5], our method is also less constrained on the trajectory, but it is not designed for the minimum-time control.

The MPC controller is given in Algorithm 8:

Algorithm 8 A centralized MPC controller

Input: $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{w}_f$, $\boldsymbol{Z}$, $\boldsymbol{Q}$
Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...

1: $k \leftarrow 0$;
2: **while** $\boldsymbol{m}_k \neq \boldsymbol{m}_f$ **do**
3: **Solve problem** (6.2);
4: **Apply** $\boldsymbol{w}_k$: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$;
5: $k \leftarrow k + 1$;
6: **end while**
7: **return** $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, ...;

$$
\begin{aligned}
\min \quad & J(\boldsymbol{m}_k, N) \\
s.t.: \quad & \boldsymbol{m}_{k+j+1} = \boldsymbol{m}_{k+j} + \Theta \cdot C \cdot \boldsymbol{w}_{k+j}, j = 0, ..., N-1 && \text{(6.2a)} \\
& \boldsymbol{G} \cdot \begin{bmatrix} \boldsymbol{w}_{k+j} \\ \boldsymbol{m}_{k+j} \end{bmatrix} \leq 0, j = 0, ..., N-1 && \text{(6.2b)} \\
& \boldsymbol{w}_{k+j} \geq 0, j = 0, ..., N-1 && \text{(6.2c)} \\
& \boldsymbol{m}_f[p_i] \geq \boldsymbol{m}_{k+j+1}[p_i] \geq \boldsymbol{m}_{k+j}[p_i], \text{ if } \boldsymbol{m}_f[p_i] \geq \boldsymbol{m}_0[p_i], && \text{(6.2d)} \\
& \qquad\qquad\qquad\qquad\qquad\qquad j = 0, ..., N-1 \\
& \boldsymbol{m}_f[p_i] \leq \boldsymbol{m}_{k+j+1}[p_i] \leq \boldsymbol{m}_{k+j}[p_i], \text{ if } \boldsymbol{m}_f[p_i] \leq \boldsymbol{m}_0[p_i], && \text{(6.2e)} \\
& \qquad\qquad\qquad\qquad\qquad\qquad j = 0, ..., N-1
\end{aligned}
$$

where $\boldsymbol{G}$ is a particular matrix deduced from the net structure and (6.2b) gives the (upper bound) constraint on firing flows to guarantee the non-negativeness of markings [64].

According to [83], the reachability space of CPN systems is a convex set. The convex subset of the reachability space $R(\mathcal{N}, \boldsymbol{m}_k, \boldsymbol{m}_f)$ corresponding to $\boldsymbol{m}_k$ and $\boldsymbol{m}_f$, inside which the following system states evolve, is generated by using constraints (6.2d) and (6.2e).

The cost function $J(\boldsymbol{m}_k, N)$ may be a linear or quadratic. For the target marking

control problem addressed here, $J(\boldsymbol{m}_k, N)$ has the quadratic form:

$$J(\boldsymbol{m}_k, N) = (\boldsymbol{m}_{k+N} - \boldsymbol{m}_f)' \cdot \boldsymbol{Z} \cdot (\boldsymbol{m}_{k+N} - \boldsymbol{m}_f) + \sum_{j=0}^{N-1} [(\boldsymbol{m}_{k+j} - \boldsymbol{m}_f)' \cdot \boldsymbol{Q} \cdot (\boldsymbol{m}_{k+j} - \boldsymbol{m}_f) \tag{6.3}$$

where weighting matrix $\boldsymbol{Z}, \boldsymbol{Q} \in \mathbb{R}^{|P|}$ are positive definite.

Example 6.2.1. *Let us consider the consistent and conservative CPN (a simple strongly connected state machine) shown in Fig.6.2(a). Since there exists only one P-semiflow in the net (the corresponding token conservation law:* $\boldsymbol{m}[p_1] + \boldsymbol{m}[p_2] + \boldsymbol{m}[p_3] = 4$*), the markings of two places are sufficient to represent the whole reachability space,* $R(\mathcal{N}, \boldsymbol{m}_0)$ *(see Fig. 6.2(b)). Let* $\boldsymbol{m}_f = [1\ 2.5\ 0.5]^T$*. The subset of its reachability space,* $R(\mathcal{N}, \boldsymbol{m}_k, \boldsymbol{m}_f)$*, generated by constraints* (6.2d) *and* (6.2e)*, is also shown (*$\boldsymbol{m}_k = \boldsymbol{m}_0$*).*

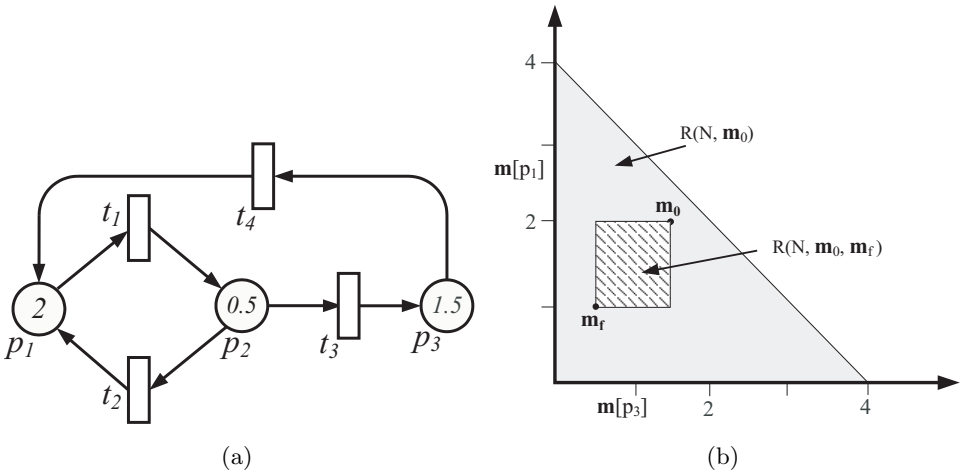


Figure 6.2: (a) A consistent and conservative CPN (a simple state machine); (b) the reachability space $R(\mathcal{N}, \boldsymbol{m}_0)$ and the subset $R(\mathcal{N}, \boldsymbol{m}_0, \boldsymbol{m}_f)$, $\boldsymbol{m}_f = [1\ 2.5\ 0.5]^T$

Proposition 6.2.2. *Let* $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *be a consistent TCPN system with* $\boldsymbol{m}_0 > 0$*, and let* $\boldsymbol{m}_f > 0$ *be a reachable final marking. By applying the MPC controller given in Algorithm 8, the closed-loop system is asymptotically stable.*

Proof: We prove the statement in two steps: 1) we prove that the problem (6.2) is feasible; 2) we define a quadratic Lyapunov function and prove that it is strictly decreasing.

1) Since the system is deterministic, i.e., noise free, it is clear that at any time step k with marking $\boldsymbol{m}_k$, one solution of problem (6.2) could be

$$\{\boldsymbol{m}_{k+j+1} = \boldsymbol{m}_k, \boldsymbol{w}_{k+j} = \boldsymbol{0}\},\ j = 0, 1, ..., N-1$$

So problem (6.2) is feasible.

2) Let $V(\boldsymbol{m}_k) = (\boldsymbol{m}_k - \boldsymbol{m}_f)^T \cdot \boldsymbol{Z} \cdot (\boldsymbol{m}_k - \boldsymbol{m}_f)$, where $\boldsymbol{Z}$ is the weight matrix in (6.2). It is clear that $V(\boldsymbol{m}_k) \geq \boldsymbol{0}$, and for any $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, $V(\boldsymbol{m}_k) \neq 0$.

Let $\boldsymbol{m}_k - \boldsymbol{m}_f = \boldsymbol{\Delta m}_k$, according to constraints (6.2d) and (6.2e) of problem (6.2), $\forall p_i \in P$, $|\boldsymbol{\Delta m}_k[p_i]| \geq |\boldsymbol{\Delta m}_{k+1}[p_i]|$. Therefore, $V(\boldsymbol{m}_k) \geq V(\boldsymbol{m}_{k+1})$.

Now we will show that $V(\boldsymbol{m}_k) > V(\boldsymbol{m}_{k+1})$ until $\boldsymbol{m}_k = \boldsymbol{m}_f$, i.e., until the system is already in the desired final state. Because the marking of each place can only move closer to its final value or stay in the same value, it is equivalent to prove that $\boldsymbol{m}_k \neq \boldsymbol{m}_{k+1}$ until $\boldsymbol{m}_k = \boldsymbol{m}_f$. Assume that at time step k, $\boldsymbol{m}_k \neq \boldsymbol{m}_f$ and $\boldsymbol{m}_k = \boldsymbol{m}_{k+1}$, i.e., at the first predictive step, the system stays at $\boldsymbol{m}_k$ with flow $\boldsymbol{w}_k = \alpha \cdot \boldsymbol{x}$ (where $\alpha \geq 0$ and $\boldsymbol{x}$ is a T-semiflow). Then, a solution of problem (6.2), Υ_1, gives a sequence of predictive states as follows:

$$\boldsymbol{m}_k \xrightarrow{\alpha \cdot \boldsymbol{x}} \begin{matrix} \boldsymbol{m}_{k+1} \\ (= \boldsymbol{m}_k) \end{matrix} \rightarrow \boldsymbol{m}_{k+2} \cdots \boldsymbol{m}_{k+N-1} \rightarrow \boldsymbol{m}_{k+N}$$

Obviously, instead of staying in $\boldsymbol{m}_k$ at the first predictive step ($\boldsymbol{m}_{k+1} = \boldsymbol{m}_k$), we may have another solution Υ_2 by starting moving to $\boldsymbol{m}_{k+2}$ at the first predictive step, and following the same sequence of states as in Υ_1, then staying in $\boldsymbol{m}_{k+N}$ at the last predictive step:

$$\boldsymbol{m}_k \rightarrow \ \boldsymbol{m}_{k+2} \ \rightarrow \boldsymbol{m}_{k+3} \cdots \boldsymbol{m}_{k+N} \xrightarrow{\alpha \cdot \boldsymbol{x}} \boldsymbol{m}_{k+N}$$

The values of cost function corresponding to Υ_1 and Υ_2 are:

$$\begin{aligned} J_1 &= \boldsymbol{\Delta m}_{k+N}^T \cdot \boldsymbol{Z} \cdot \boldsymbol{\Delta m}_{k+N} + \boldsymbol{\Delta m}_k^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_k \\ &\quad + \sum_{j=1}^{N-1} \boldsymbol{\Delta m}_{k+j}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_{k+j} \end{aligned} \tag{6.4}$$

$$\begin{aligned} J_2 &= \boldsymbol{\Delta m}_{k+N}^T \cdot \boldsymbol{Z} \cdot \boldsymbol{\Delta m}_{k+N} + \boldsymbol{\Delta m}_k^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_k \\ &\quad + \sum_{j=2}^{N} \boldsymbol{\Delta m}_{k+j}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_{k+j} \end{aligned} \tag{6.5}$$

Therefore, the following can be obtained:

$$\begin{aligned} J_2 - J_1 &= \boldsymbol{\Delta m}_{k+N}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_{k+N} - \boldsymbol{\Delta m}_{k+1}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_{k+1} \\ &= \boldsymbol{\Delta m}_{k+N}^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_{k+N} - \boldsymbol{\Delta m}_k^T \cdot \boldsymbol{Q} \cdot \boldsymbol{\Delta m}_k \end{aligned}$$

According to the constraints (6.2d) and (6.2e), if $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, we must have $\boldsymbol{m}_{k+N} \neq \boldsymbol{m}_k$ and $\forall p_i \in P$, $|\boldsymbol{\Delta m}_k[p_i]| \geq |\boldsymbol{\Delta m}_{k+N}[p_i]|$. Otherwise, the system should stay in $\boldsymbol{m}_k$ at every predictive step, but obviously this cannot be an optimal solution: since the net is consistent, $\boldsymbol{m}_k > 0$ (inside $R(\mathcal{N}, \boldsymbol{m}_0, \boldsymbol{m}_f)$) and all transitions are

controllable, the state marking can move in any direction [97]; in particular, it is possible to move towards $\boldsymbol{m}_f$, hence it is possible to decrease the distance to the desired marking in one step. Therefore, $J_2 - J_1 < 0$, i.e., Υ_2 is a better solution than Υ_1, implying that if $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, by applying the MPC controller given in Algorithm 8 the system always starts moving towards $\boldsymbol{m}_f$ from the first predictive step. So $V(\boldsymbol{m}_k) > V(\boldsymbol{m}_{k+1})$ until $\boldsymbol{m}_k = \boldsymbol{m}_f$. ■

Example 6.2.3. *Let us still consider the net system shown in Fig 6.2(a) with the final state* $\boldsymbol{m}_f = [1\ 2.5\ 0.5]^T$. *Assume that the firing rates of all the transitions are equal to* 0.1 *and the sampling period* $\Theta = 0.1$. *By applying the MPC controller shown in Algorithm 8 (with* $\boldsymbol{Q} = \boldsymbol{I}, \boldsymbol{Z} = 1000 \cdot \boldsymbol{I}, N = 5$*), the obtained marking trajectory of places* p_1 *and* p_3 *is illustrated in Fig. 6.3 (in continuous line), and the close-loop cost is 99793. We also apply to the system the MPC controller proposed in [64] (with* $\boldsymbol{Q} = \boldsymbol{I}, \boldsymbol{Z} = 1000 \cdot \boldsymbol{I}, \boldsymbol{R} = \boldsymbol{0}$*), in which the state trajectory follows a straight line from* $\boldsymbol{m}_0$ *to* $\boldsymbol{m}_f$ *(in dotted line), and the close-loop cost is equal to 114220 that is larger than by using the method proposed here.*

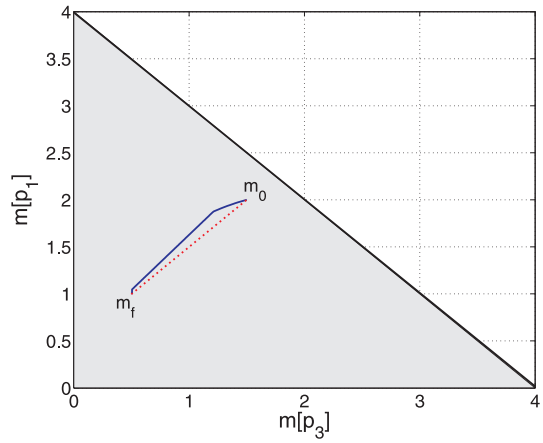


Figure 6.3: Marking trajectory of the net system in Fig. 6.2(a) by applying: the MPC controller proposed in [64] (dotted line) and the MPC controller shown in Algorithm 8 (continuous line)

6.3 Application to Distributed MPC control

Let us consider now a (large scale) PN system that is composed of a set of subsystems denoted by K. Those subsystems $S^l = \langle \mathcal{N}^l, {\boldsymbol{m}_0}^l \rangle \in K$ are connected with places modelling buffers denoted by B. The transitions connecting with buffer places are said to be *interface* transitions. Let P^l, T^l be the sets of places and transitions of subsystem S^l. The partition of the system is described as follows:

- $\bigcap_{S^l \in K} P^l = \emptyset$, $(\bigcup_{S^l \in K} P^l) \cup B = P$;

- $\bigcap_{S^l \in K} T^l = \emptyset$, $\bigcup_{S^l \in K} T^l = T$;
- for any place $p_i \in B$, ${}^\bullet p_i \cap P^l \neq \emptyset \Rightarrow {}^\bullet p_i \subseteq P^l$ and $p_i{}^\bullet \cap P^l \neq \emptyset \Rightarrow p_i{}^\bullet \subseteq P^l$, i.e., buffer places are input and output private.

In the sequel, the following notations are used:

- $B^{(l,k)} = \{p_i \in B | {}^\bullet p_i \subseteq S^l, p_i{}^\bullet \subseteq S^k\}$ is the set of output buffers of S^l and input buffers of S^k;
- $B^{(\cdot,l)} = \bigcup_{S^k \in K} B^{(k,l)}$ and $B^{(l,\cdot)} = \bigcup_{S^k \in K} B^{(l,k)}$;
- $C^l \in \mathbb{N}^{|P^l| \times |T^l|}$ is the flow matrix of subsystem S^l.

In the distributed setting, each subsystem is controlled independently by the MPC controller given in Algorithm 8. Therefore the states of each subsystem S^l will be interior points (inside $R(\mathcal{N}^l, \boldsymbol{m}_0{}^l, \boldsymbol{m}_f{}^l)$. However, one key issue we need to consider is that the markings of buffer places may be keeping decreased and converging to zero if the same control laws are applied (in TCPNs under infinite server semantics, once a place is marked it takes infinite to empty it). Since we consider the control in finite time, in the sequel, if $\boldsymbol{m}_k[p_i] \leq \epsilon_1$ with ϵ_1 a small positive value, we assume that under the actual control law the marking of the buffer place p_i is converging to zero. When some buffer places with their markings smaller than ϵ_1, subsystems may "stop", because the required flows to move towards the final states may be constrained by these buffer places. We will design a particular strategy to put more tokens to these places with their markings converging to zero.

6.3.1 Two subsystems

For clarity, let us first consider the system composed of only two subsystems. Later, the control method is directly extended to the case of multiple subsystems.

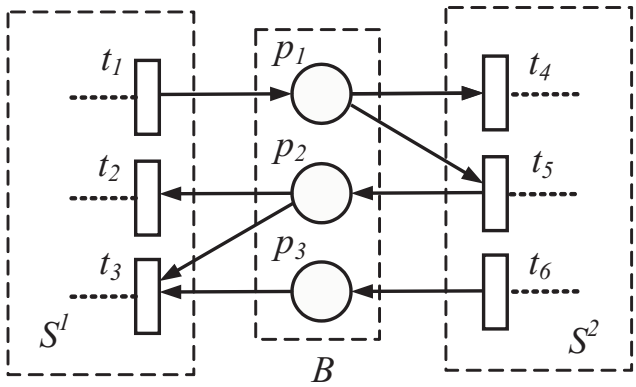


Figure 6.4: A distributed CPN composed of two subsystems

Fig. 6.4 shows the interface transitions and buffer places of a CPN that is composed of two subsystems S^1 and S^2, we have $B^{(1,2)} = B^{(1,\cdot)} = \{p_1\}$, $B^{(2,1)} = B^{(2,\cdot)} = \{p_2, p_3\}$.

Let us denote by $\boldsymbol{m}_0{}^1$ and $\boldsymbol{m}_0{}^2$ the initial states (markings) of subsystems S^1 and S^2, while the desired final states are denoted by $\boldsymbol{m}_f{}^1$ and $\boldsymbol{m}_f{}^2$. The initial and final states of the global system, including the buffer places, are $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$. Instead of a global central controller, each subsystem will have its own local MPC controller. A local controller has only information about the structure and state of its corresponding subsystem, as well as the connected buffers places. The problem we address here is to drive the subsystems S^1 and S^2 to their desired final states. We use the following assumptions:

(A1) The global system is consistent (therefore, each subsystem is also consistent).

(A2) The initial state of the global system $\boldsymbol{m}_0 > 0$ (including the buffer places); $\boldsymbol{m}_f > 0$ is a given reachable final state of the global system, and $\boldsymbol{m}_f{}^1, \boldsymbol{m}_f{}^2 > 0$ are the corresponding final states of subsystems (observe that we do not require *liveness* or *deadlock-freeness*).

(A3) In the untimed (autonomous) model, subsets of buffer places never define a siphon that can be emptied.

Notice that although in assumption (A2) we assume a given final state of the global system, we are only focusing on driving subsystems to their corresponding final states. Regarding the buffer places, we simply ensure that they are always in legal (non-negative) states.

Assumption (A3) can be checked in the following way. Let us define $\mathbf{Pre}_\Sigma$ and $\mathbf{Post}_\Sigma$ as $|P| \times |T|$ sized matrices for a given net system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ such that:

- $\mathbf{Pre}_\Sigma[p,t] = |t^\bullet|$ if $\mathbf{Pre}[p,t] > 0$, $\mathbf{Pre}_\Sigma[p,t] = 0$ otherwise
- $\mathbf{Post}_\Sigma[p,t] = 1$ if $\mathbf{Post}[p,t] > 0$, $\mathbf{Post}_\Sigma[p,t] = 0$ otherwise.

Equations $\{\boldsymbol{y}^T \cdot \boldsymbol{C}_\Sigma \leq \mathbf{0},\ \boldsymbol{y} \geq \mathbf{0}\}$ where $\boldsymbol{C}_\Sigma = \mathbf{Post}_\Sigma - \mathbf{Pre}_\Sigma$ define a generator of siphons (Σ is a siphon iff $\exists \boldsymbol{y} \geq \mathbf{0}$ such that $\Sigma = \|\boldsymbol{y}\|$, $\boldsymbol{y}^T \cdot \boldsymbol{C}_\Sigma \leq \mathbf{0}$) [28, 87]. Hence, the following system:

$$\begin{array}{ll} \bullet\ \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma},\ \ \boldsymbol{m}, \boldsymbol{\sigma} \geq 0, & \{\text{state equation}\} \\ \bullet\ \boldsymbol{y}^T \cdot \boldsymbol{C}_\Sigma \leq \mathbf{0}, \boldsymbol{y} \geq \mathbf{0}, & \{\text{siphon generator}\} \\ \bullet\ \boldsymbol{y}^T \cdot \boldsymbol{m} = 0, & \{\text{empty siphon at } \boldsymbol{m}\} \end{array} \tag{6.6}$$

has no solution iff the continuous net system has no emptied siphon. We only need to solve problem (6.6) off-line considering the part of system composed of the buffer places and interface transitions, denoted by S': because every firing sequence that can fire in the original system S can also fire in S' and, normally S' has much smaller

size than the whole system. Nevertheless, in some particular cases, it may be not necessary to solve problem (6.6). For example, given a EQ net, we can easily check its consistency and conservativeness, then by applying the rank theorems we can verify the liveness the boundedness in polynomial time. If the net system is live and bounded, then we can directly conclude that there exists no emptied siphon.

It is clearly that the buffer places may constrain the flows of the interface transitions. Thus, the following additional constraint to problem (6.2) should be added for each subsystem S^l:

$$\boldsymbol{w}_k^l[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_k[p_i]}{\boldsymbol{Pre}[p_i,t]},\ \forall t \in p_i{}^\bullet, p_i \in B^{(\cdot,l)} \tag{6.7}$$

where $\boldsymbol{w}_k^l[t]$ is the flow of transition t and $\boldsymbol{m}_k[p_i]$ is the marking of buffer place p_i at time step k. Then, for each subsystem S^l, the modified problem that should be solved at every time step k is:

$$\begin{aligned}
\min \quad & J(\boldsymbol{m}_k^l, N) && \text{(6.8a)}\\
s.t.: \quad & \boldsymbol{m}_{k+j+1}^l = \boldsymbol{m}_{k+j}^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_{k+j}^l, j = 0, ..., N-1 && \text{(6.8b)}\\
& \boldsymbol{G}^l \cdot \begin{bmatrix} \boldsymbol{w}_{k+j}^l \\ \boldsymbol{m}_{k+j}^l \end{bmatrix} \leq 0,\ j = 0, ..., N-1 && \text{(6.8c)}\\
& \boldsymbol{w}_{k+j}^l \geq 0,\ j = 0, ..., N-1 && \text{(6.8d)}\\
& \boldsymbol{m_f}^l[p_i] \geq \boldsymbol{m}_{k+j+1}^l[p_i] \geq \boldsymbol{m}_{k+j}^l[p_i],\ \text{if } \boldsymbol{m_f}^l[p_i] \geq \boldsymbol{m_0}^l[p_i], && \text{(6.8e)}\\
& \qquad\qquad\qquad\qquad\qquad\qquad j = 0, ..., N-1 &&\\
& \boldsymbol{m_f}^l[p_i] \leq \boldsymbol{m}_{k+j+1}^l[p_i] \leq \boldsymbol{m}_{k+j}^l[p_i],\ \text{if } \boldsymbol{m_f}^l[p_i] \leq \boldsymbol{m_0}^l[p_i], && \text{(6.8f)}\\
& \qquad\qquad\qquad\qquad\qquad\qquad j = 0, ..., N-1 &&\\
& \boldsymbol{w}_k^l[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_k[p_i]}{\boldsymbol{Pre}^l[p_i,t]}, \forall t \in p_i{}^\bullet, p_i \in B^{(\cdot,l)} && \text{(6.8g)}
\end{aligned}$$

If all the (input) buffer places of subsystem S^l are marked, according to Proposition 6.2.2, problem (6.8) is also feasible, because we can always fire a small $\boldsymbol{w}_k^l \geq 0$ such that constraint (6.8g) is not active. Moreover, if at every time step k all of its input buffer places are marked, using the same reasoning as in Proposition 6.2.2, we will have $\boldsymbol{m}_k^l \neq \boldsymbol{m}_{k+1}^l$, so $V(\boldsymbol{m}_k^l) > V(\boldsymbol{m}_{k+1}^l)$; therefore, the subsystem keeps evolving until the final state is reached.

Nevertheless, if the markings of some of the input buffer places of a subsystem S^l are converging to zero (with their markings smaller than ϵ_1), S^l may "stop" in certain state before reaching $\boldsymbol{m_f}^l$. In this case, S^l has to wait on the current state until its neighboring subsystem puts more tokens into these buffer places. At the same time, some T-semiflows might be fired in S^l, putting more tokens into its output buffer places; since these output buffer places of S^l are inputs buffer places of its neighboring subsystem, consequently this will help the evolution of its neighboring

subsystem to the final state. Therefore, when S^l "stops" evolving we will use another cost function, denoted by $H(\boldsymbol{m}_k^l)$:

$$H(\boldsymbol{m}_k^l) = \sum_{\forall t_j \in {}^\bullet p_i, p_i \in B^{(l,\cdot)}, \boldsymbol{m}_k[p_i] \leq \epsilon_1} -\boldsymbol{w}_k^l[t_j] \tag{6.9}$$

where $\boldsymbol{m}_k[p_i]$ is the marking of buffer place p_i at time step k. By minimizing $H(\boldsymbol{m}_k^l)$, we try to put more tokens into its output buffer places p_i, $\boldsymbol{m}_k[p_i] \leq \epsilon_1$ (if there exist) and it is equivalent to maximizing their input transition flows; but meanwhile, it may also try to empty its input buffer places. Therefore, in order to keep certain amounts of tokens inside a marked input buffer place p_i, the following constrains are added:

$$\Theta \cdot \sum_{t_j \in p_i{}^\bullet} \boldsymbol{w}_k^l[t_j] \cdot \boldsymbol{Pre}[p_i, t_j] \leq \alpha \cdot \boldsymbol{m}_k[p_i], \ \forall p_i \in B^{(\cdot,l)} \tag{6.10}$$

where $0 \leq \alpha < 1$. It mean that by one step, the marking of a marked buffer place p_i can be maximally decreased to $(1-\alpha) \cdot \boldsymbol{m}_k[p_i]$. When cost function $H(\boldsymbol{m}_k^l)$ is applied, we fix the time horizon $N = 1$, then problem (6.8) is modified to (6.11):

$$\begin{aligned}
\min \quad & H(\boldsymbol{m}_k^l) && \text{(6.11a)}\\
s.t.: \quad & \boldsymbol{m}_{k+1}^l = \boldsymbol{m}_k^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_k^l && \text{(6.11b)}\\
& \boldsymbol{G}^l \cdot \begin{bmatrix} \boldsymbol{w}_k^l \\ \boldsymbol{m}_k^l \end{bmatrix} \leq 0 && \text{(6.11c)}\\
& \boldsymbol{w}_k^l \geq 0 && \text{(6.11d)}\\
& {\boldsymbol{m}_f}^l[p_i] \geq \boldsymbol{m}_{k+1}^l[p_i] \geq \boldsymbol{m}_k^l[p_i], \text{ if } {\boldsymbol{m}_f}^l[p_i] \geq {\boldsymbol{m}_0}^l[p_i] && \text{(6.11e)}\\
& {\boldsymbol{m}_f}^l[p_i] \leq \boldsymbol{m}_{k+1}^l[p_i] \leq \boldsymbol{m}_k^l[p_i], \text{ if } {\boldsymbol{m}_f}^l[p_i] \leq {\boldsymbol{m}_0}^l[p_i] && \text{(6.11f)}\\
& \boldsymbol{w}_k^l[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_k^l[p_i]}{\boldsymbol{Pre}^l[p_i, t]}, \forall t \in p_i{}^\bullet, p_i \in B^{(\cdot,l)} && \text{(6.11g)}\\
& \Theta \cdot \sum_{t_j \in p_i{}^\bullet} \boldsymbol{w}_k^l[t_j] \cdot \boldsymbol{Pre}[p_i, t_j] \leq \alpha \cdot \boldsymbol{m}_k[p_i], \forall p_i \in B^{(\cdot,l)} && \text{(6.11h)}
\end{aligned}$$

The procedure of the distributed MPC controller consists of solving problem (6.8) and/or problem (6.11) in each subsystem S^l at any time step: if the final state ${\boldsymbol{m}_f}^l$ has already been reached, problem (6.11) is solved, trying to put more tokens to its output buffers with their markings converging to zero (remember that S^l stays in ${\boldsymbol{m}_f}^l$ because of constraints (6.11e) and (6.11f)); otherwise, problem (6.8) is first solved and, if subsystem S^l is able to evolve towards its final state (still inside the convex $R(\mathcal{N}^l, \boldsymbol{m}_0^l, \boldsymbol{m}_f^l)$), the first predictive control law is applied; if by solving (6.8) the system stops in $\boldsymbol{m}_k$, then problem (6.11) is solved. Because one subsystem may reach the final state faster than the other, each subsystem should communicate to its neighbors when its final state has been reached. This procedure repeats until the

final states of both subsystems have been reached. The local controller of subsystem S^l is given in Algorithm 9.

Algorithm 9 Distributed MPC control: algorithm for subsystem S^l

Input: S^l, $\boldsymbol{Z}$, $\boldsymbol{Q}$, ${\boldsymbol{m}_f}^l$, α
Output: $\boldsymbol{w}_0^l$, $\boldsymbol{w}_1^l$, $\boldsymbol{w}_2^l$, ...

$k \leftarrow 0$;
while $\exists S^i, \boldsymbol{m}_k^i \neq {\boldsymbol{m}_f}^i$, $i = 1, 2$ **do**
  if $\boldsymbol{m}_k^l = {\boldsymbol{m}_f}^l$ **then**
    Solve problem (6.11)
  else
    Solve problem (6.8)
    if $\boldsymbol{m}_k^l = \boldsymbol{m}_{k+1}^l$ **then**
      Solve problem (6.11)
    end if
  end if
  Apply $\boldsymbol{w}_k^l$: $\boldsymbol{m}_{k+1}^l \leftarrow \boldsymbol{m}_k^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_k^l$
  Update the states of buffers
  $k \leftarrow k + 1$
end while
return $\boldsymbol{w}_0^l$, $\boldsymbol{w}_1^l$, $\boldsymbol{w}_2^l$, ...

Remark 6.3.1. *As we have already mentioned, in TCPNs under infinite server semantics it will take infinite time to empty a marked place, therefore the initially marked buffer places cannot totally get emptied in finite time and thus the subsystem will not be totally stopped. Hence, in the implementation of Algorithm 9 we approximate condition "*$\boldsymbol{m}_k^l = \boldsymbol{m}_{k+1}^l$*" (implying that* S^l *stops in* $\boldsymbol{m}_k^l$*) by using "*$(\boldsymbol{m}_k^l - \boldsymbol{m}_{k+1}^l)^T \cdot (\boldsymbol{m}_k^l - \boldsymbol{m}_{k+1}^l) \leq \epsilon_2$*", where* ϵ_2 *is a small positive value.*

Proposition 6.3.2. *Let* $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ *be TCPN system composed of two subsystems* $\mathcal{S}^l = \langle \mathcal{N}^l, \boldsymbol{\lambda}^l, {\boldsymbol{m}_0}^l \rangle$, $l = 1, 2$. *If assumptions (A1) to (A3) are satisfied, by applying Algorithm 9, each subsystem* S^l *converges to its corresponding final state* ${\boldsymbol{m}_f}^l$ *in finite time.*

Proof: If all the buffer places are marked, according to Proposition 6.2.2 we can find a solution of problem (6.8) such that the obtained state of the next step $\boldsymbol{m}_{k+1}^l \neq \boldsymbol{m}_k^l$, then $V(\boldsymbol{m}_{k+1}^l) < V(\boldsymbol{m}_k^l)$, i.e., subsystem S^l evolves towards ${\boldsymbol{m}_f}^l$. If a subsystem "stops" in a state $\boldsymbol{m}_k^l \neq \boldsymbol{m}_f^l$, it is because some buffer places p_i are converging to zero ($\boldsymbol{m}_k[p_i] \leq \epsilon_1$). We will prove that by using the proposed algorithm, these buffer places can get marked, and the subsystem will keep evolving towards the final state.

Without loss of generality, assume that subsystem S^1 has stopped in a state before reaching ${\boldsymbol{m}_f}^1$, then problem (6.11) should be solved in S^1. Consider now

subsystem S^2, there are two cases: (i) S^2 is able to keep evolving towards ${\boldsymbol{m}_f}^2$ by solving problem (6.8); (ii) S^2 also stops in a state before reaching ${\boldsymbol{m}_f}^2$. In case (i) the final state of S^2 will be reached in finite time, then problem (6.11) should be solved; in case (ii), problem (6.11) should also be solved according to the algorithm. Therefore, we need to prove that by solving (6.11) in both subsystems, these buffer places with their marking converging to zero, will get marked.

Assume that by applying Algorithm 9 the system has "stopped" at $\boldsymbol{m}_k$, then problem (6.11) should be solved in both subsystems. According to assumption (A3), there exists no empty siphon composed of buffer places. If we consider each subsystem independently, its states are forced to be inside the closed interior convex subset $R(\mathcal{N}^l, \boldsymbol{m}_0^l, \boldsymbol{m}_f^l)$, so both subsystems have positive markings. Therefore, there exists no empty siphon in the (global) system. On the other hand, since the net is consistent, the system is possible to move in any direction [97]. In particular, because $\boldsymbol{m}_f > 0$ is reachable from $\boldsymbol{m}_k$, there must exist a global flow such that at the next step some buffer places p_i, $\boldsymbol{m}_k[p_i] \leq \epsilon$ get marked.

Now let us consider the subsystems. Clearly, a place can only get tokens by means of firing its input transitions. Therefore, by solving problem (6.11) in which we try to maximize the input transition flows of buffer places p_i with $\boldsymbol{m}_k[p_i] \leq \epsilon$, some of these places will get marked. Hence, once both subsystems solve problem (6.11) and the obtained control laws are applied, at least one of these buffer place will be marked. At the same time, because of constrains (6.11h) (with $0 \leq \alpha < 1$), for any already marked buffer place p_j, its marking can be maximally decreased to $(1-\alpha) \cdot \boldsymbol{m}_k[p_j] > 0$. By repeating this process, more buffer places with markings converging to zero will be marked (until all of them are marked, if necessary); for any already marked buffer places p_j, its marking can be maximally decreased to $(1-\alpha)^n \cdot \boldsymbol{m}_k[p_j]$, where n is the (bounded) number of buffer places. By choosing an appropriated α and a small enough positive number $\epsilon_1 < (1-\alpha)^n \cdot \boldsymbol{m}_k[p_j]$, it means that all the buffer places are marked. After that, both subsystems can keep evolving towards the final state by solving problem (6.8). ■

6.3.2 Multiple subsystems

Algorithm 9 can be directly applied to the distributed control of a system composed of multiple subsystems and the convergence to the final states could be proved using a similar argument as in Proposition 6.3.2. Let us point out that, one subsystem may have multiple neighbors, hence have multiple sets of buffer places and interface transitions; and those related constraints (in problem (6.8) and (6.11)) should be applied to all of them.

On the other hand, let us address the ending condition of Algorithm 9 (step 2). As we have already mentioned, one subsystem may reach its final state faster than the others. However, the algorithm (executed in each subsystem) finishes only if all the subsystems have reached their final states. The reason is very clear: one subsystem that has already been in its final state may still need to put more tokens to its output buffer places (by firing some T-semiflows), which are required

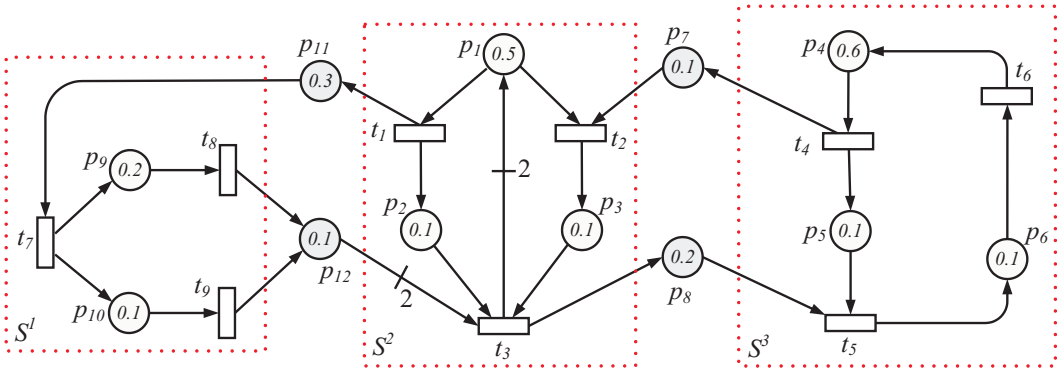


Figure 6.5: A simple TCPN example composed of 3 subsystems

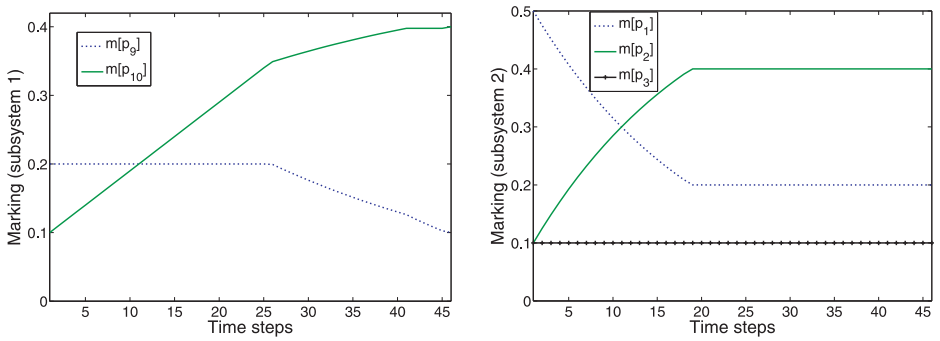
by its neighboring subsytems. Therefore, when a subsystem has reached its final state, it should propagate this information to all the other subsystems, through the connections among neighbors.

6.3.3 A simple example

Let us consider the consistent and conservative net system shown in Fig. 6.5, which can be seen as composed of three subsystems. The input and output buffer places of S^1 are $B^{(\cdot,1)} = \{p_{11}\}$ and $B^{(1,\cdot)} = \{p_{12}\}$, respectively; the input and output buffer places of S^2 are $B^{(\cdot,2)} = \{p_7, p_{12}\}$ and $B^{(2,\cdot)} = \{p_8, p_{11}\}$, respectively; meanwhile, the input and output buffer places of S^3 are $B^{(\cdot,3)} = \{p_8\}$ and $B^{(3,\cdot)} = \{p_7\}$, respectively.

The initial state $\boldsymbol{m}_0$ is shown in Fig. 6.5, and we assume that the desired final state (including the buffer places) is $\boldsymbol{m}_f = [0.2\ 0.4\ 0.1\ 0.2\ 0.4\ 0.2\ 0.1\ 0.5\ 0.1\ 0.4\ 0.4\ 0.3]^T$, the firing rate of transition t_4 is 1; while for all the other transitions the firing rate is equal to 0.5. The sampling period $\Theta = 0.1$. Clearly, the net system is not CF (e.g., conflicts appear in p_1), therefore, the decentralized control method proposed in Chapter 5 is not applicable; at the same time, we can easily verify that not all subsystems are mono-T-semiflow, e.g. the net in S^1 is not conservative, so the control method proposed in [4] may not be applicable either. It can be checked that there exists no siphon composed of buffer places and the net is consistent. By applying the distributed MPC controller proposed in this chapter, the state evolution of each subsystem and buffers are shown in Fig. 6.6 (obtained by using time horizon $N = 5$, $\alpha = 0.5$, $\epsilon_1 = 0.01$ and $\epsilon_2 = 10^{-6}$).

All the subsystems reach their final states asymptotically, but, not at the same time instant. It can be observed that subsystem S^3 reaches its final state faster (after 17 time steps); then subsystem S^2 reaches its final state (after 19 time steps); subsystem S^1 reaches its final state slowest (after 46 time steps). However, the markings of buffers places have not reached the values specified in $\boldsymbol{m}_f$. For instance, $\boldsymbol{m}_f[p_7] = 0.1$ and $\boldsymbol{m}_f[p_8] = 0.5$, but by using the distributed MPC controller the marking of place p_7 reaches 0.5 and the marking of place p_8 reaches 0.1. Finally,



(a) state evolution of S^1

(b) state evolution of S^2

(c) state evolution of S^3

(d) state evolution of buffers

Figure 6.6: The state evolution of the net system in Fig. 6.5 controlled by DMPC

let us point out that the states of each subsystem S^l are constrained to be inside the subset of reachability space $R(\mathcal{N}^l, \boldsymbol{m}_0{}^l, \boldsymbol{m}_f{}^l)$. Hence, if a place p_i has its initial marking $\boldsymbol{m}_0[p_i] = \boldsymbol{m}_f[p_i]$, then the marking of p_i will remain constant during the whole trajectory, for example place p_3 of subsystem S^2.

6.4 Conclusions

In this chapter we present a distributed method for the target marking control problem of general TCPNs. Similarly to previous methods, we also assume that subsystems are connected by sets of (buffers) places. Each subsystem is controlled by a local controller, which is able to access all the local variables as well as the buffer places connected to the corresponding subsystem. We first propose a centralized MPC controller, in which the state of system is forced to be inside an interior convex subset of the reachability space and asymptotic stability is guaranteed. Then, this MPC controller is applied in a distributed setting. We present a distributed control algorithm, in which two optimization problems should be solved in an interleaved way: one is similar to that in the centralized MPC, and another one is used to "recover" from situations where the markings of some buffer places are converging

to zero (consequently, subsystems may stop evolving towards the final state). We prove that by using this algorithm, the final (positive) states of subsystems can be reached in finite time. Meanwhile, for the buffer places, we ensure that they are always in legal (non-negative) states.

In the proposed control method, we do not need a coordinator as in the approach presented in Chapter 5. However, once a subsystem has reached its final state, it must transmit this information to all the other subsystems because the algorithm stops only if all the subsystems reach their final states. Although this information should be propagated to all the subsystems (through the connections among neighbors), the amount of data transmitted is small, therefore the communication cost is still low.

Finally, let us point out that, if one is interested in minimum-time decentralized control and the net is CF, the method proposed in Chapter 5 (a reduction technique is employed) can be applied. If the net is not CF, this method is no longer applicable, but we can use the distributed MPC proposed in this chapter for general nets. However, it is not designed for minimum-time control.

Chapter 7

Minimum-time Flow Control of CF nets

This chapter discusses the (optimal) flow control problem of TCPNs. Instead of driving the system to a given desired final marking (as we have discussed in chapters 4, 5 and 6, from both the centralized and decentralized point of view), here we try to reach an optimal flow in minimum-time. In other words, we generalize from reaching a final state to reaching a "final region" (with the objective of minimum-time evolution). In particular, we assume CF net systems and we are interested in driving the system as fast as possible to a steady state (belonging to a convex region) where the maximal flow is obtained. The main difference from the target marking control problem, also the main challenge, is that we may not be able to uniquely determine a desired final state, therefore the control methods proposed in the previous chapters are not applicable directly. We propose a heuristic algorithm, in which at each time step we first compute an estimated "best" firing count vector that drives the system to the convex region where the maximal flow is obtained; then an ON/OFF strategy is applied. Later, we show that some additional firings can further decrease the time spent to reach the maximal flow.

7.1 Motivations

Optimal flow control problems are widely studied using different system models such as Petri nets, queueing networks, etc., (see, for example, [55, 78, 109]). In [55] a control design for CPN was proposed, trying to obtain the flow minimising the cost function composed by production cost, stocking cost, ordering cost and break-up cost. Contribution [78] studied the optimal flow control policies for a stochastic fluid-flow network; it aims to minimize the total expected discounted cost defined by the reward for admission of fluid into the buffer and the cost incurred for holding fluid in the buffer. The work in [109] proposed two flow control algorithms for networks with multiple paths between each source/destination pair, both are distributed algorithms over the network to maximize aggregate source utility, which can be described as a function of transmission rates. In this work, we focus on the optimal flow control of CF net systems, addressing the problem of reaching an optimal flow from a given initial state, while *minimizing the time* spent on the trajectory.

The optimal steady-state control problem of CPNs has been addressed in [65], trying to *maximize a profit function* depending on the marking in the steady-state ($\boldsymbol{m}_{ss}$), the (controlled) flow in the steady state ($\boldsymbol{w}_{ss}$, $\boldsymbol{C} \cdot \boldsymbol{w}_{ss} = \boldsymbol{0}$), and the initial marking ($\boldsymbol{m}_0$). Here, we assume that the profit function is aiming to maximize the flow (under certain constrains on control inputs) of steady state for a given $\boldsymbol{m}_0$. Since only one minimal T-semiflow exists in strongly connected and consistent CF nets [93], it is equivalent to maximize the flow of any transition t_j, by means of the following LPP [65]:

$$\begin{array}{rl} \psi_j = & \max \boldsymbol{w}_{ss}[t_j] \\ \text{s.t.} & \boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_{ss} = \boldsymbol{0} \\ & \boldsymbol{w}_{ss}[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_{ss}[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^\bullet t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_{ss}, \boldsymbol{\sigma}, \boldsymbol{m}_{ss} \geq \boldsymbol{0} \end{array} \tag{7.1}$$

where $\boldsymbol{v}[p_i, t]$ are *slack* variables.

Usually the solution of LPP (7.1) is not unique (different $\boldsymbol{m}_{ss}$ may exist for a given $\boldsymbol{w}_{ss}$). Let us denote by $\psi_j = \boldsymbol{w}_{ss}[t_j]$ the optimal flow of transition t_j obtained by solving (7.1), and $\mathcal{M}$ the set of markings with the maximal flow, i.e.,

$$\begin{array}{l} \mathcal{M} = \{\boldsymbol{m} | \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \boldsymbol{\sigma} \geq 0 \text{ and } \exists 0 \leq \boldsymbol{u} \leq \boldsymbol{f}, \\ \qquad \boldsymbol{w} = \boldsymbol{f} - \boldsymbol{u}, \boldsymbol{C} \cdot \boldsymbol{w} = \boldsymbol{0}, \boldsymbol{w}[t_j] = \psi_j\} \end{array} \tag{7.2}$$

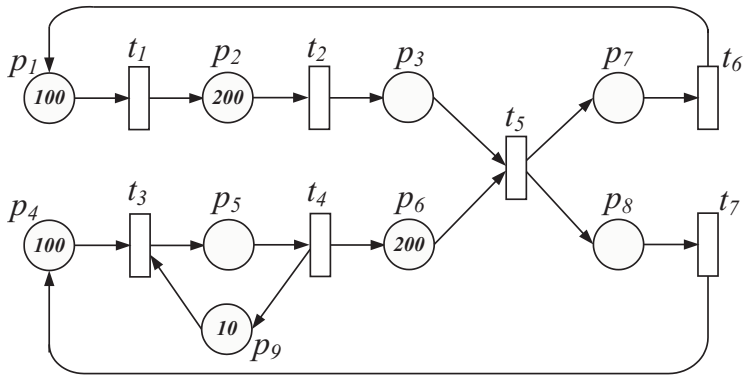
where $\boldsymbol{f}$ is the (uncontrolled) flow vector at marking $\boldsymbol{m}$. Any state in $\mathcal{M}$ is an equilibrium point corresponding to the maximal flow that can be maintained by applying an appropriate control $\boldsymbol{u}$. Because all the constrains of (7.2) are linear, $\mathcal{M}$ is a convex subset included in the reachability space. It arises an interesting problem: which state $\boldsymbol{m} \in \mathcal{M}$ can be reached in minimum-time (by applying appropriate control methods)? or equivalently, how the maximal flow can be obtained in minimum-time? Here we call this problem *Minimum-time Flow Control* problem.

7.2 Difficulties of Minimum-time Flow Control

We already know that for CF nets, given a final state and the corresponding minimal firing count vector, the minimum-time control strategy is ON/OFF (proposed in Section 4.2). However, in the Minimum-time Flow Control problem, we do not know which firing count vector (thus the marking) is the one that minimizes the time spent on the trajectory. In particular, the time spent is not monotonic with respect to the corresponding firing count vectors.

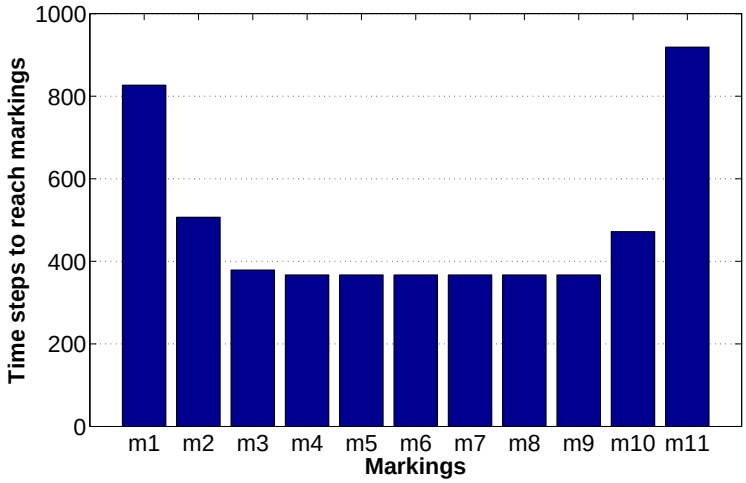
Let us consider the MG (a subclass of CF nets) in Fig.7.1 and assume that sampling period $\Theta = 0.01$. By solving LPP (7.1), we obtain that the maximal flow of any transition (because in MGs $\mathbf{1}$ is the unique T-semiflow) is $\psi = 1$. Two of the possible steady-states (in the same region) corresponding to the maximal flow are: $\boldsymbol{m}_1 = [100\ 170\ 20\ 94\ 6\ 190\ 10\ 10\ 4]^T$ and $\boldsymbol{m}_{11} = [100\ 2\ 188\ 94\ 6\ 190\ 10\ 10\ 4]^T$, $\boldsymbol{m}_1, \boldsymbol{m}_{11} \in \mathcal{M}$, with corresponding minimal firing count vectors $\boldsymbol{\sigma}_1 = [0\ 30\ 6\ 0\ 10\ 0\ 0]^T$ and $\boldsymbol{\sigma}_{11} = [0\ 198\ 6\ 0\ 10\ 0\ 0]^T$, respectively. Obviously, $\boldsymbol{\sigma}_1 \leq \boldsymbol{\sigma}_{11}$. Let us consider 9 intermediate points on the straight line from $\boldsymbol{m}_1$ to $\boldsymbol{m}_{11}$, obtained by $\boldsymbol{m}_i = (1-\alpha)\cdot\boldsymbol{m}_1 + \alpha\cdot\boldsymbol{m}_{11}$, $i = 2, 3, ..., 10, \alpha = 0.1, 0.2, ..., 0.9$. Intermediate markings $\boldsymbol{m}_2$ to $\boldsymbol{m}_{10}$ belong to $\mathcal{M}$, hence they are also steady-states with the optimal flow, and the corresponding minimal firing count vectors satisfy $\boldsymbol{\sigma}_1 \leq \boldsymbol{\sigma}_2 \leq ... \leq \boldsymbol{\sigma}_{10} \leq \boldsymbol{\sigma}_{11}$. By applying the ON/OFF controller, the numbers of time steps for reaching $\boldsymbol{m}_1$ to $\boldsymbol{m}_{11}$ starting from $\boldsymbol{m}_0 = [100\ 200\ 0\ 100\ 0\ 200\ 0\ 0\ 10]^T$ are shown in Fig.7.2.

Figure 7.1: A simple MG with the maximal flow $\psi = 1$, firing rate vector $\boldsymbol{\lambda} = [1\ 0.5\ 0.25\ 1/6\ 0.05\ 0.1\ 0.1]^T$



For reaching $\boldsymbol{m}_1$ by applying the ON/OFF controller, the required number of time steps is 827. For $\boldsymbol{m}_2$, it is decreased to 507 (remember that $\boldsymbol{\sigma}_1 \leq \boldsymbol{\sigma}_2$). The required number of time steps is further decreased to 379 for $\boldsymbol{m}_3$ to $\boldsymbol{m}_9$. But it starts to increase from $\boldsymbol{m}_{10}$. For reaching $\boldsymbol{m}_{11}$, 919 time steps are required. We can easily observe that a smaller $\boldsymbol{\sigma}$ does not provide less time to obtain the maximal flow. Furthermore, the non-monotonicity respect to the firing count vector has been exhibited in this example.

Figure 7.2: The time steps required to reach different steady-states with the maximal flow by applying the ON/OFF controller to the MG shown in Fig.7.1: non-monotonicity appears with respect to the corresponding firing count vectors



The difference between $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, for example, is that in $\boldsymbol{\sigma}_2$ transition t_2 fires more than in $\boldsymbol{\sigma}_1$. Therefore, p_3 receives more tokens and t_5 may fire faster (its flow is increased). In the cases of $\boldsymbol{\sigma}_1$ to $\boldsymbol{\sigma}_9$, transition t_5 is the one that fires "slowest", i.e., the one that requires more time steps to fire the given firing amount. Therefore by increasing the flow of t_5, the overall number of time steps is decreased. On the other hand, if t_2 fires too much, as in $\boldsymbol{\sigma}_{10}$ and $\boldsymbol{\sigma}_{11}$, t_2 becomes the one that requires more time steps, so the overall time steps starts to increase.

7.3 A heuristic algorithm for CF nets

In a (strongly connected and consistent) CF net there exists a unique minimal T-semiflow $\boldsymbol{x}$ and its *support* contains all the transitions [93]. Therfore, if ψ_j is the maximal flow of transition t_j (the optimal solution of LPP (7.1)), then the maximal flow of every transition can be deduced (because $\boldsymbol{\psi}$ is a steady-state flow, $\boldsymbol{C} \cdot \boldsymbol{\psi} = 0$ and $\boldsymbol{\psi} = \alpha \cdot \boldsymbol{x}$, $\alpha > 0$). Moreover, the minimal required marking of a place p_i to ensure the maximal flow can be easily determined by the firing rate of its unique output transition and weight on the arc:

Definition 7.3.1. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a CF system, $\boldsymbol{x}$ be the minimal T-semiflow and ψ_j be the optimal flow of transition t_j. Then, $\boldsymbol{\mu}$ is said to be the minimal required marking for the optimal flow* [1]*, if:*

$$\boldsymbol{\mu}[p_i] = (\psi_j / \boldsymbol{\lambda}[t]) \cdot (\boldsymbol{x}[t] / \boldsymbol{x}[t_j]) \cdot \boldsymbol{Pre}[p_i, t], \forall p_i \in P, \{t\} = p_i^{\bullet} \tag{7.3}$$

[1] It is a marking vector that may not be reachable.

An immediate consequence of Definition 7.3.1 is the following: given $\boldsymbol{m} \geq \boldsymbol{\mu}$, the uncontrolled flow of any transition t_j corresponding to $\boldsymbol{m}$ satisfies $\boldsymbol{f}[t_j] \geq \psi_j$. Therefore, if all the transitions are controllable, there exists $0 \leq \boldsymbol{u} \leq \boldsymbol{f}$, such that the controlled flow $\boldsymbol{w}[t_j] = \psi_j$. In other words, for any reachable marking $\boldsymbol{m}$, $\boldsymbol{m} \in \mathcal{M}$ iff $\boldsymbol{m} \geq \boldsymbol{\mu}$. Thus, the Minimum-time Flow Control problem of CF nets is equivalent to *reaching a marking* $\boldsymbol{m} \geq \boldsymbol{\mu}$ *in minimum-time.* Moreover, a firing count vector $\boldsymbol{\sigma}$ that leads to a steady state $\boldsymbol{m}_{ss}$ with the maximal flow is a solution of the following equations:

$$\begin{aligned} &\boldsymbol{m}_{ss} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ &\boldsymbol{m}_{ss} \geq \boldsymbol{\mu} \\ &\boldsymbol{\sigma} \geq 0 \end{aligned} \tag{7.4}$$

As we have discussed in Chatper 4, the ON/OFF controller is a minimum-time controller of CF nets assuming a given firing count vector. However, the firing count vector satisfying (7.4) is not unique in general. Therefore, we need to compute the best one, i.e., the one that leads to the maximal flow in minimum-time by applying the ON/OFF controller.

According to the ON/OFF strategy, every transition fires as fast as possible, until each one completes its required amount given by the corresponding firing count vector. Therefore, the overall time is determined by the "slowest" transition, i.e., the one that costs most time steps to fire its given amount. Since the firing speed is variable in TCPN under infinite server semantics, depending on the state evolution, we will consider an *estimation of the number of time steps* (something similar to the concept we have used in the B-ON/OFF controller (see Section 4.4.2)).

Let us assume that the current marking at time step k is $\boldsymbol{m}_k$ and let $\boldsymbol{\sigma}_k$ be a firing count vector that should be fired to reach a state in $\mathcal{M}$. Then $\boldsymbol{S}_k[t_j] = \lceil \frac{\boldsymbol{\sigma}_k[t_j]}{\lambda_j \cdot enab(\boldsymbol{m}_k, t_j) \cdot \Theta} \rceil$ can be viewed as an estimation of the number of time steps that transition t_j needs to fire (it is an estimation because it is assumed a constant speed for t_j). Given transitions t_a and t_b, if $\boldsymbol{S}_k[t_a] > \boldsymbol{S}_k[t_b]$, then it would be said that t_a is "slower" than t_b. Notice that $\boldsymbol{S}_k$ does not give neither a lower nor an upper bound because $\boldsymbol{m}_k$ changes dynamically.

In the heuristics we propose, at each time step k we minimize the number of time steps of the slowest transition, i.e., to minimize the infinity norm of $\boldsymbol{S}_k$, $||\boldsymbol{S}_k||_\infty = \max\{|\boldsymbol{S}_k[t_j]|\}, t_j \in T$. The minimization of $||\boldsymbol{S}_k||_\infty$ can be done by solving the following LPP, in which a new variable d is introduced:

$$\begin{aligned} \mathit{min} \quad & d \\ \text{s.t.} \quad & \boldsymbol{m}_{ss} = \boldsymbol{m}_k + \boldsymbol{C} \cdot \boldsymbol{\sigma}_k \\ & \boldsymbol{m}_{ss} \geq \boldsymbol{\mu} \\ & d \geq \boldsymbol{\sigma}_k[t]/(\boldsymbol{\lambda}[t] \cdot enab(\boldsymbol{m}_k, t) \cdot \Theta), \forall t \in T \\ & \boldsymbol{\sigma}_k \geq \boldsymbol{0} \end{aligned} \tag{7.5}$$

where $\boldsymbol{m}_k$ is a the current marking at time step k.

The control progress is given in Algorithm 10, which is a close-loop control and at each time step we recompute the "best" firing count vector $\boldsymbol{\sigma}_k$ according to the current state. Then, $\boldsymbol{\sigma}_k$ is fired by applying the ON/OFF strategy, obtaining a heuristics for the Minimum-time Flow Control. Since the stability of applying the ON/OFF controller to CF nets has been proved, the convergence of Algorithm 10 can be easily obtained.

Algorithm 10 An algorithm of Minimum-time Flow Control problem for CF nets

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, t_j, Θ

Output: sequence of controlled flows: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$

1: compute ψ_j by solving LPP (7.1);
2: compute $\boldsymbol{\mu}$ that satisfies (7.3);
3: $k \leftarrow 0$;
4: **while** not $(\boldsymbol{m}_k \geq \boldsymbol{\mu})$ **do**
5: compute $\boldsymbol{\sigma}_k$ by solving LPP (7.5);
6: compute the controlled flow $\boldsymbol{w}_k$ corresponding to the ON/OFF strategy;
7: update state: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$;
8: $k \leftarrow k + 1$;
9: **end while**
10: compute the steady state controlled flow $\boldsymbol{w}_k$, such that: $\boldsymbol{C} \cdot \boldsymbol{w}_k = 0, \boldsymbol{w}_k[t_j] = \psi_j$;
11: **Return** $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$;

Algorithm 10 can be further improved, considering the *persistency* property of CF nets: the (additional) firing of one transition does not disable the firing of the others [93]; however, it may increase the flow of the other transitions (as in the net system in Fig.7.1, additional firings of t_2 increased the flow of t_5). Based on this observation and because our problem is to drive the system to a marking $\boldsymbol{m} \in \mathcal{M}$, i.e., $\boldsymbol{m} \geq \boldsymbol{\mu}$, for any transition t, if at time step k all of its input place $p_i \in {}^\bullet t$ satisfy $\boldsymbol{m}_k[p_i] > \boldsymbol{\mu}[p_i]$, we can fire t without increasing the time to reach $\mathcal{M}$. So, in the improved algorithm we distinguish the following two cases:

(1) for any transition t with $\boldsymbol{\sigma}_k[t] = 0$, we consider the markings of the input places of t: if for any $p_i \in {}^\bullet t$, $\boldsymbol{m}_k[p_i] > \boldsymbol{\mu}[p_i]$, then t is fired as fast as possible; else, t is blocked;

(2) for any transition t with $\boldsymbol{\sigma}_k[t] > 0$ the ON/OFF strategy is applied.

The strategy of case (1) can only decrease the time for reaching a marking in $\mathcal{M}$, but not increase. This is because we would fire t only if all of its input places already have *more-than-enough* markings to obtain the maximal flow; at the same time this firing will not "slow down", but may "speed up", the firing of others. This improved process is given in Algorithm 11.

Proposition 7.3.2. *Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be a CF net system. By applying Algorithm 11, the system converges to a steady-state $\boldsymbol{m} \in \mathcal{M}$ that maximizes the flow.*

Algorithm 11 Improved algorithm of Minimum-time Flow Control problem for CF nets

Input: $\langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$, t_j, Θ
Output: sequence of controlled flows: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$

compute ψ_j by solving LPP (7.1);
compute $\boldsymbol{\mu}$ that satisfies (7.3);
$k \leftarrow 0$;
compute $\boldsymbol{\sigma}_k$ by solving LPP (7.5);
while not $(\boldsymbol{m}_k \geq \boldsymbol{\mu})$ **do**
  for all $t \in T$ **do**
    if $\boldsymbol{\sigma}_k[t] > 0$ **then**
      compute the controlled flow $\boldsymbol{w}_k[t]$ corresponding to the ON/OFF strategy;
    else if $\boldsymbol{m}_k[p_i] > \boldsymbol{\mu}[p_i]$ for any $p_i \in {}^\bullet t$ **then**
      $\boldsymbol{w}_k[t] \leftarrow \boldsymbol{\lambda}[t] \cdot enab(\boldsymbol{m}_k, t)$;
    else
      $\boldsymbol{w}_k[t] \leftarrow 0$;
    end if
  end for
  update state: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$;
  $k \leftarrow k + 1$;
end while
compute the steady state controlled flow $\boldsymbol{w}_k$, such that: $\boldsymbol{C} \cdot \boldsymbol{w}_k = 0, \boldsymbol{w}_k[t_j] = \psi_j$;
Return $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$;

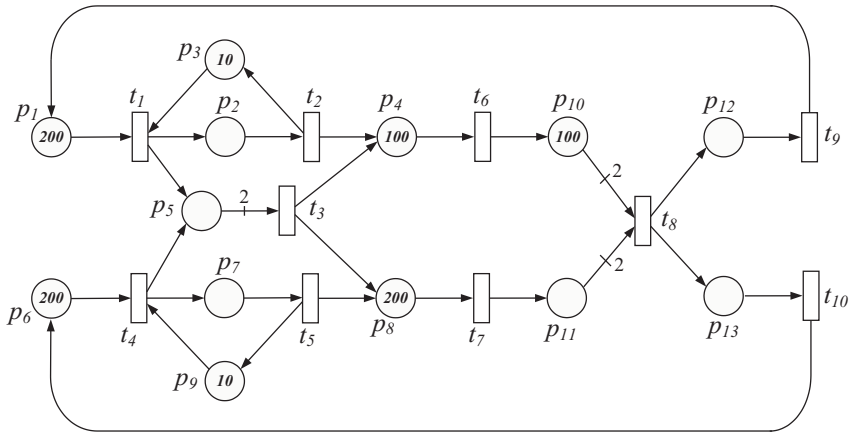
Proof: Since $\mathcal{N}$ is a CF net, the additional firings (for a transition t with $\boldsymbol{\sigma}_k[t] = 0$) do not disable the firing of $\boldsymbol{\sigma}_k$ that drives the system to a state $\boldsymbol{m} \in \mathcal{M}$. On the other hand, for any place p_i with $\boldsymbol{m}[p_i] \leq \boldsymbol{\mu}[p_i]$, we do not decrease its marking, therefore the algorithm will converge to a marking $\boldsymbol{m}' \geq \boldsymbol{\mu}$ belonging to $\mathcal{M}$. ■

Algorithm 11 is still a heuristics for minimum-time. One clear reason is that we try to look for the "best" firing count vector (in terms of spending less time on firing it) based on an approximation of the time steps that is obtained from the current state and flow; nevertheless, the risk of choosing a very "bad" one is somehow reduced because after each time step we recompute it based on the actual state. Another possible reasons is that only *local* information is considered. By means of some firings, the time spent for reaching a marking in $\mathcal{M}$ may be decreased. But, in the case concerning a transition t with $\boldsymbol{\sigma}_k[t] = 0$, it is allowed to fire t again only if its input places have tokens more than those in $\boldsymbol{\mu}$. However, this strategy may not be the optimal in some situations, even for MGs (a simple subclass of CF nets, see the net in Fig. 7.5 for a example).

7.4 Examples

Let us consider the CF net system in Fig.7.3, assuming $\Theta = 0.01$. The unique minimal T-semiflow of the net is $\boldsymbol{x} = [1\ 1\ 1\ 1\ 1\ 2\ 2\ 1\ 1\ 1]^T$.

Figure 7.3: A CF net system with the maximal flow $\psi_9 = 1$, firing rate vector $\boldsymbol{\lambda} = [0.25\ 1/6\ 0.05\ 0.25\ 1/6\ 0.1\ 0.5\ 0.05\ 1/30\ 1/30]^T$

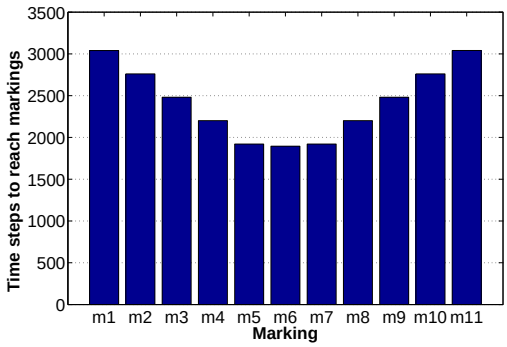


Assume that we want to maximize the flow of transition t_9, by solving LPP (7.1) with $t_j = t_9$, it is obtained $\psi_j = 1$. From (7.3), the corresponding minimal required marking is $\boldsymbol{\mu} = [4\ 6\ 4\ 20\ 40\ 4\ 6\ 4\ 4\ 40\ 40\ 30\ 30]^T$. However, the solution of LPP (7.4) is not unique. For instance, $\boldsymbol{\sigma}_1 = [34\ 28\ 0\ 6\ 0\ 0\ 100\ 30\ 0\ 0]^T$ and $\boldsymbol{\sigma}_{11} = [6\ 0\ 0\ 34\ 28\ 0\ 100\ 30\ 0\ 0]^T$ are both solutions of LPP (7.4), reaching optimal-flow steady states $\boldsymbol{m}_1 = [166\ 6\ 4\ 128\ 40\ 194\ 6\ 100\ 4\ 40\ 40\ 30\ 30]^T$ and $\boldsymbol{m}_{11} = [194\ 6\ 4\ 100\ 40\ 166\ 6\ 128\ 4\ 40\ 40\ 30\ 30]^T$. Similarly to the example of the MG in Fig.7.1, we also consider 9 more intermediate points on the straight line from $\boldsymbol{m}_1$ to $\boldsymbol{m}_{11}$ and the maximal flow can be obtained from all of them. The time steps required for reaching $\boldsymbol{m}_i$, $i = 1, 2, ..., 11$, by using the ON/OFF controller, the results of applying Algorithm 10 and Algorithm 11, are illustrated in Fig.7.4.

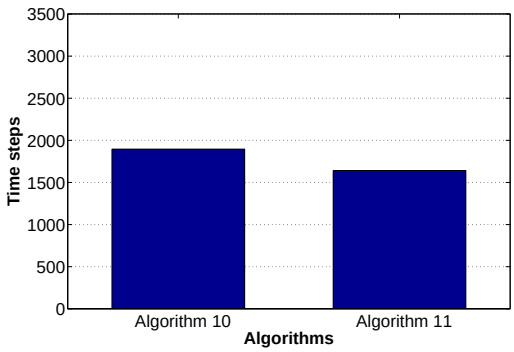
As shown in Fig. 7.4, by applying Algorithm 10 we can obtain the maximal flow in 1895 time steps, which is the same as using the ON/OFF controller to drive the system to $\boldsymbol{m}_6$. However, we should remember that we do not know *a priori* that driving the system to $\boldsymbol{m}_6$ will cost less time than to other markings $\boldsymbol{m}_i$, $i = 1, 2, ..., 11$, $i \neq 6$. By applying Algorithm 11, the time to reach the maximal flow is further reduced to 1641 time steps.

Although Algorithm 11 can highly reduce the time spent for reaching a marking in $\mathcal{M}$, the minimum-time is not guaranteed in general, even for MGs. Let us consider a MG shown in Fig.7.5, assuming that the firing rate vector $\boldsymbol{\lambda} = [1\ 1\ 1\ 1\ 1/3\ 1\ 1]^T$ and $\Theta = 0.01$. The maximal flow is $\psi = 1$ (obtained by solving LPP (7.1)) and the corresponding minimal required marking is $\boldsymbol{\mu} = [1\ 1\ 1\ 3\ 3\ 1\ 1\ 1]^T$. By using Algorithm 10, we can reach the maximal flow in 138 time steps. By Algorithm 11, we can reduce the number of time steps to 102, reaching steady-state $\boldsymbol{m} = [5.64\ 2.941$

Figure 7.4: Comparison of time steps for reaching the maximal of the CF system in Fig.7.3



(a) time steps required to reach different steady-states with the maximal flow by applying the ON/OFF control



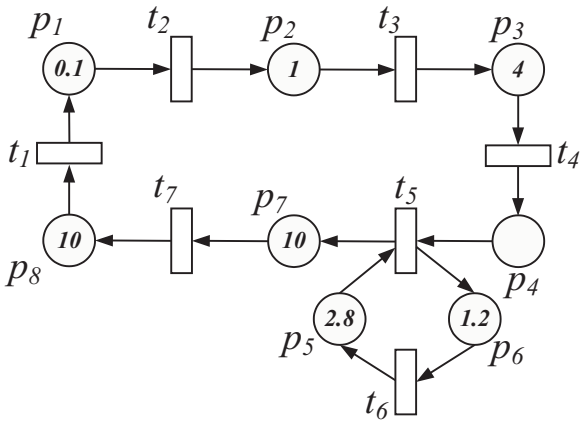
(b) time steps required to reach the maximal flow by applying the proposed algorithms

2.609 3 3 1 3.587 7.323]^T and the corresponding firing count vector $\boldsymbol{\sigma}$ = [9.09 3.55 1.609 3 0 0.2 6.413]^T. Nevertheless, it is still not the minimum-time for reaching the maximal flow: by firing $\boldsymbol{\sigma'}$ = [8.942 3.437 1.598 3 0 0.2 6.34]^T using the ON/OFF strategy, we reach another maximal flow steady-state $\boldsymbol{m'}$ = [5.605 2.84 2.598 3 3 1 3.66 7.398]^T in only 100 time steps.

7.5 Conclusions

In this chapter we discuss Minimum-time (Optimal) Flow Control problems for CF net systems. The main difference from the target marking control problem (at the same time the main difficulty of solving it) is that, in general we cannot uniquely determine a steady state with the given optimal flow (in our case the maximal flow), and actually, they belong to a convex region. Then, two issues arise: which steady state with maximal flow can be reached fastest? and which control method should

Figure 7.5: A MG example where Algorithm 11 does not give the minimum-time to the maximal flow



be applied? We have already known that the ON/OFF controller is a minimum-time controller assuming a given firing count vector, and here we first focus on how to choose the "best" one. We propose a heuristic algorithm for CF nets, in which we compute and update at each time step the "best" firing count vector according to an estimation of the number of time steps for firing. We also show that by means of some additional firings (because of the persistency of CF nets, the firing of one transition will reduce the enabling degrees of other transitions, but may increase their flows), the time to reach the maximal flow can be further reduced. Concerning the computational complexity, in each time step we solve a LPP, therefore, in polynomial time.

Chapter 8

Simulations and Comparisons

The control methods proposed in this thesis have been implemented and integrated to SimHPN, a Matlab toolbox for hybrid PNs [48]. In this chapter we carry out several case studies using the SimHPN toolbox for control laws computation and for simulations. In the first three case studies we focus on the centralized control methods and the last one illustrates the distributed control.

8.1 Implementation: SimHPN

The control methods proposed in this thesis are implemented and integrated into a Matlab embedded toolbox for hybrid Petri nets, called SimHPN. It provides a collection of tools for simulation, analysis and synthesis of dynamical systems that are modelled by continuous, discrete and hybrid PNs. Different firing server semantics are supported for both continuous and discrete transitions, as infinite server semantics and product server semantics. Moreover, deterministic delays with single server semantics are also available for discrete transitions. Besides of simulation options, SimHPN also offers some useful tools such as computing the structural elements (P/T-semiflows), performance bounds, optimal steady-state control; the optimal observability and diagnosis of continuous models. Both the data related to the model description, i.e., the net structures, markings, timing parameters etc., and the output results can be exported to the Matlab workspace and then used for further analysis.

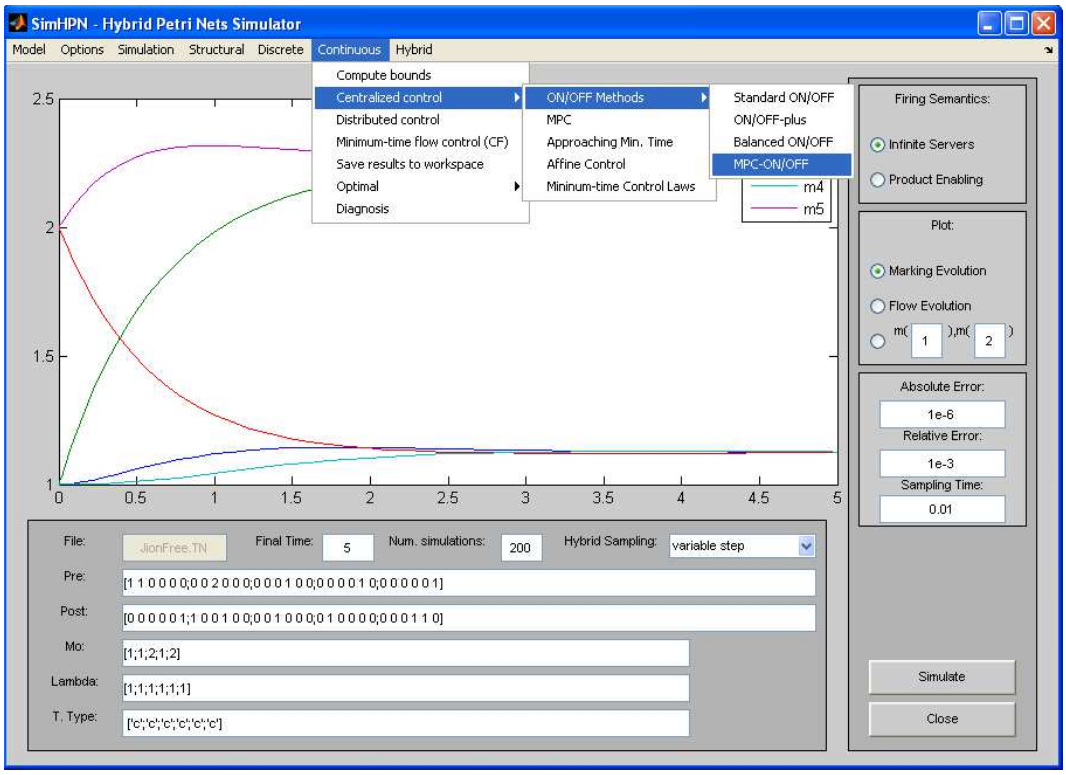


Figure 8.1: The main Graphical User Interface of SimHPN

The main GUI (Graphical User Interface) of the toolbox is shown in Fig. 8.1. The model description, i.e., $\boldsymbol{Pre}$, $\boldsymbol{Post}$, $\boldsymbol{\lambda}$ and $\boldsymbol{m}_0$, can be imported from other PN editors such as *Pmeditor* or *TimeNet* [53], or from a *.mat* file; alternatively, users can also directly input the parameters through the edit boxes.

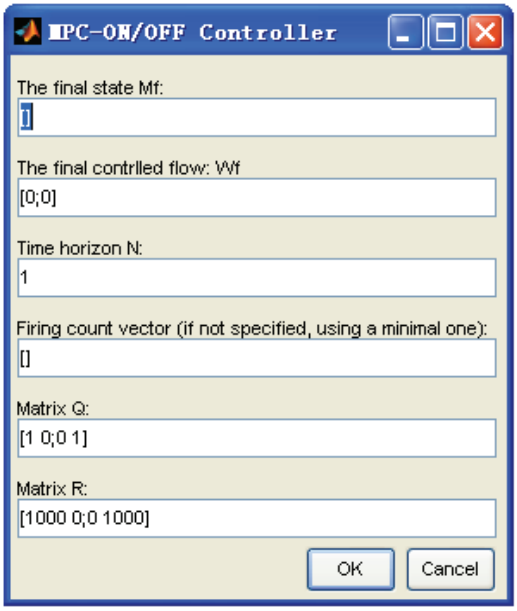


Figure 8.2: The pop-up window corresponding to the MPC-ON/OFF controller

Regarding the control of TCPNs, we can choose the menu "continuous" from the Menu Bar, then in the pop-up sub-menus, users can select an appropriate controller, both centralized or distributed are available. For the target marking control, the desired final state $\boldsymbol{m}_f$ and other parameters (if exist) can be input through edit boxes. For instance, by selecting the MPC-ON/OFF controller from the centralized method, a window as in Fig. 8.2 appears. In the case that distributed control is selected, users need to provide the number of subsystems. Then, a window shown in Fig. 8.3 appears, in which the definition of subsystems should be inputted: for each subsystem, its buffer places and interface transitions should be given. In the corresponding edit box, the first line should be a row vector composed of the index of buffer places, and the second line should be a row vector composed of the index of interface transitions. For example, Fig. 8.3 gives the input parameters related to the example of three subsystems shown in Fig. 6.5. Users can also choose the minimum-time flow control for CF nets, in which we automatically compute the maximal flow of the system and give a heuristic minimum-time control for reaching the flow.

The toolbox is available at http://webdiis.unizar.es/GISED/?q=tool/simhpn, and more technical details can be found in [48].

In the sequel, we consider several examples from flexible manufacturing systems and Automatic Guided Vehicle Systems (AGVS). We simulate those centralized control methods and compare the results (time steps and CPU time) by using the first 3 examples. Apart from the methods proposed in this thesis, another heuristics

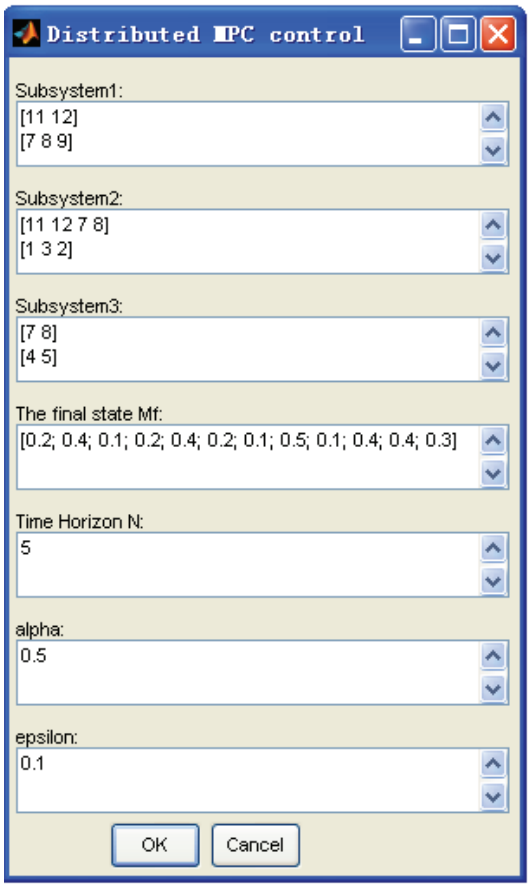
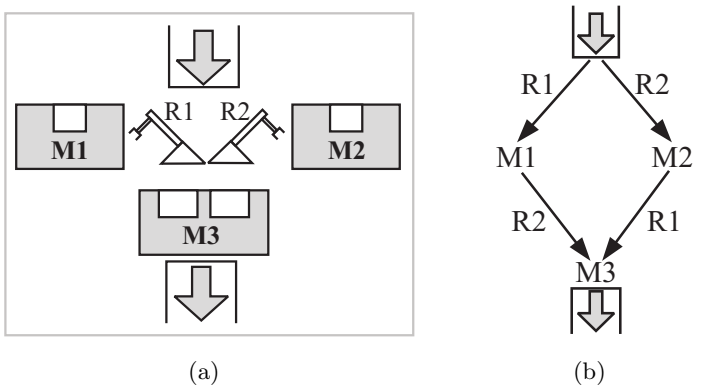


Figure 8.3: The pop-up window corresponding to the distributed control

for minimum-time control, the approaching minimal-time controller proposed in [5], is also included in the comparison. In the last example, decentralized control methods are considered. The simulations are performed by using Matlab 8.0 on a PC with Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz, 3.24GB of RAM.

8.2 Case study 1: centralized control of a manufacturing cell

Let us consider the manufacturing cell shown in Fig. 8.4. It consists of three machines M1, M2, M2 and two robots R1, R2. Two semi-products A and B are processed by M1 and M2, respectively, then they are assembled in M3 to get the final product. Robot R1 moves the raw materials from the input buffer to M1 or moves the semi-product B from M2 to M3; robot R2 moves the materials from the input buffer to M2 or moves the semi-product A from M1 to to M3. The logical layout and production process are given in Fig. 8.4(a) and Fig. 8.4(b).



(a) (b)

Figure 8.4: (a) Logical layout of a manufacturing cell (b) Its production process

The PN model of the described manufacturing cell is presented in Fig. 8.5. When machine M1 is available (p_{14} is marked) and robot R1 is idle (p_{18} is marked), a part of raw material for semi-product A can be loaded to machine M1, changing to loading state (p_1). When the loading process finishes (t_2 fires), M1 changes to working state (p_2) and robot R1 is freed (p_{18} is marked). Transition t_3 models the working process of machine M1, and when it finishes, the semi-product A is stored in p_3. If the slots for semi-product A in machine M3 (p_{12}) is available and robot R2 is idle, the semi-product A can be loaded from machine M1 to machine M3 then waits in p_9. The behavior of M2 for processing semi-product B is modelled in a similar way, but the robots are used in a different order. Finally, if both semi-products A (in p_9) and B (in p_{10}) are available, they are assembled (*rendez-vous*) to the final product. The meaning of the places and transitions of the PN model is in Table 8.1.

We assume that each robot can only handle one piece of raw materials or semi-products and machine M1, M2 can accept maximally two pieces at the same time; machine M3 has 4 free slots for each type of semi-products and can assemble 2 pairs of semi-products at the same time; and it is assumed that we initially have 5 pieces of raw materials. According to this setting, the initial state $\boldsymbol{m}_0$ of the system is described in Fig. 8.5. Let us point out that, in this system there exist deadlocks; however, the system can be driven to any reachable final state by using appropriate control methods.

Considering the system as timed, we assume that every transition t_j has an average delay time, denoted by $\boldsymbol{\delta}[t_j]$. In particular, the loading time of raw materials to machine M1 and M2 are all equal to 0.1 time units, i.e., $\boldsymbol{\delta}[t_1] = \boldsymbol{\delta}[t_2] = \boldsymbol{\delta}[t_6] = \boldsymbol{\delta}[t_7] = 0.1$; the loading operations of semi-products from machines M1 to M3 and M2 to M3 take 0.4 time units, i.e., $\boldsymbol{\delta}[t_4] = \boldsymbol{\delta}[t_5] = \boldsymbol{\delta}[t_9] = \boldsymbol{\delta}[t_{10}] = 0.4$; the processing of machine M1 requires 0.5 time units, and for M2, it is 0.8 time units, i.e., $\boldsymbol{\delta}[t_3] = 0.5, \boldsymbol{\delta}[t_8] = 0.8$; it takes 0.4 time units to combine the two semi-products, and 2 time units to process them in machine M3, i.e., $\boldsymbol{\delta}[t_{11}] = 0.4, \boldsymbol{\delta}[t_{12}] = 2$. In the corresponding TCPN model under infinite server semantics, time delays are ap-

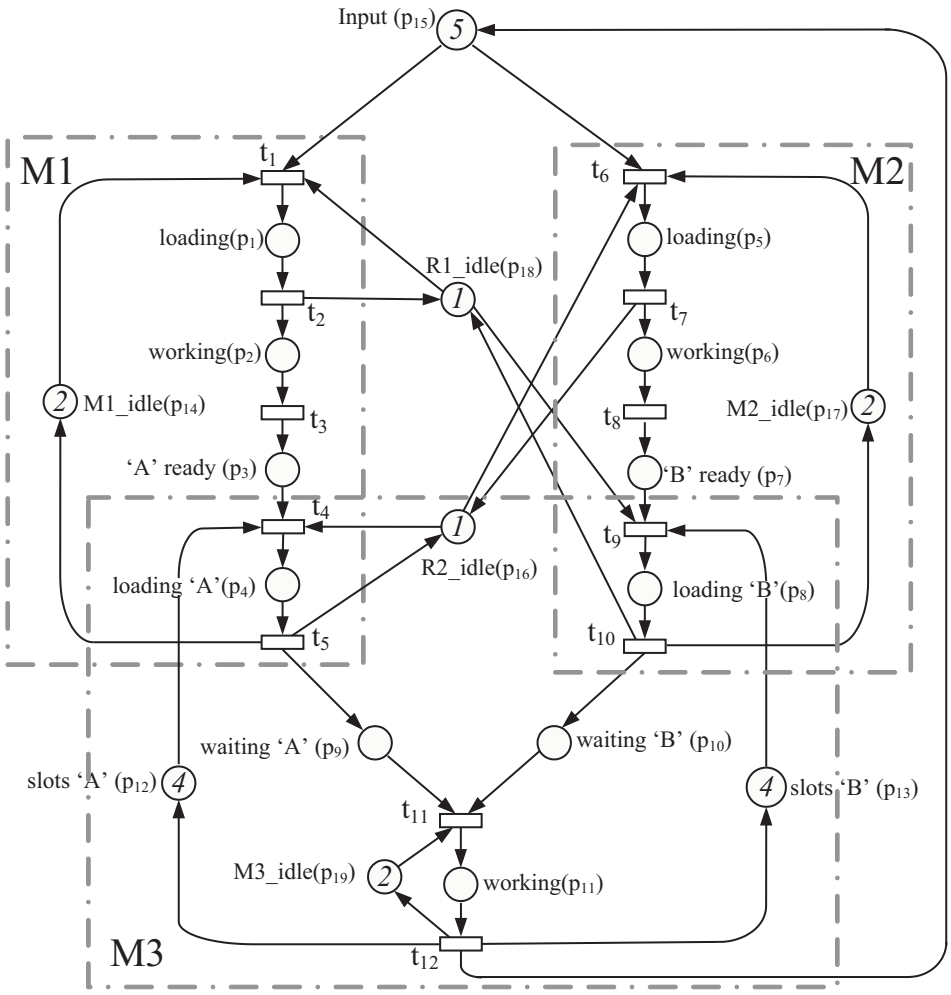


Figure 8.5: The PN model of a manufacturing cell, where robots R1 and R2 are shared resources: R1 is used to move raw materials into M1, and to move semi-products from M2 to M3; R2 is used to move raw materials into M2, and to move semi-products from M1 to M3.

proximated by their mean values ($\boldsymbol{\lambda}[t_j] = 1/\boldsymbol{\delta}[t_j]$, $t_j \in T$), obtaining a deterministic approximation of the discrete case [80].

Let us consider a target marking control problem of this system. In order to have a positive initial marking, we assume that all the empty places in Fig. 8.5 have initial marking equal to 0.1. For a manufacturing system, normally we want to maximize the throughput (flow) of the system. In particular, we can verify that this system has a unique minimal T-semiflow equal to $\mathbf{1}$; therefore, it is equivalent to maximize the flow of any transition $t_j \in T$. By solving the same LPP as in (7.1) (a simple optimal steady-state control problem [65]), the maximal flow is $\psi = 0.775$.

With $\psi = 0.775$, we can compute a final marking state $\boldsymbol{m}_f$ with the minimal

Table 8.1: The interpretation of the PN model in Fig. 8.5

Place	Interpretation	Transition	Interpretation
p_1	M1 is loading	t_1	R1 starts to load M1
p_2	M1 is working	t_2	R1 loading finishes
p_3	semi-product A is ready	t_3	M1 finishes processing
p_4	M3 is loading A	t_4	R2 starts to load A to M3
p_5	M2 is loading	t_5	R2 loading A finishes
p_6	M2 is working	t_6	R2 starts to load M2
p_7	semi-product B is ready	t_7	R2 loading finishes
p_8	M3 is loading B	t_8	M2 finishes processing
p_9	A is waiting for assembling	t_9	R1 starts to load B to M3
p_{10}	B is waiting for assembling	t_{10}	R1 loading B finishes
p_{11}	M3 is working	t_{11}	combine A and B
p_{12}	available slots for A	t_{12}	assembling finishes
p_{13}	available slots for B		
p_{14}	M1 is available		
p_{15}	input raw materials		
p_{16}	R2 is idle		
p_{17}	M2 is available		
p_{18}	R1 is idle		

Work In Process (WIP) cost, by solving the a similar LPP problem:

$$\begin{array}{ll} \min & \boldsymbol{l} \cdot \boldsymbol{m}_f \\ \text{s.t.} & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_f = \boldsymbol{0} \\ & \boldsymbol{w}_f[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t],\ \forall t \in T, p_i \in {}^\bullet t \\ & \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_f[t_j] = \psi,\ \forall t_j \in T \\ & \boldsymbol{w}_f, \boldsymbol{\sigma}, \boldsymbol{m}_f \geq \boldsymbol{0} \end{array} \tag{8.1}$$

where $\boldsymbol{l}$ is the cost vector due to immobilization to maintain the production flow, e.g., due to the levels in stores. In this example, let us assume that we try to minimize the storage, i.e., the number of tokens, in the buffer places, i.e., $\boldsymbol{l}[p_3] = \boldsymbol{l}[p_7] = \boldsymbol{l}[p_9] = \boldsymbol{l}[p_{10}] = 1$ and for other places p_i, let $\boldsymbol{l}[p_i] = 0$. By solving LPP (8.1), we obtain an optimal final state $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.8125\ 1.083\ 0.8125\ 0.55]^T$, such that the maximal flow and minimal WIP cost are achieved.

We also consider a variant of the net system in Fig. 8.5, shown in Fig. 8.6. The difference is that now the robots R1, R2 are used in a different manner: R1 is shared by machine M1 and M2 to move raw materials into machines; while R2 is shared by

machine M1 and M2 to move semi-products into M3. The same initial marking and firing rates are used as before and by solving the same LPPs, we obtain the maximal flow at a final marking with the minimal WIP cost: $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.58\ 1.083\ 1.045\ 0.55]^T$.

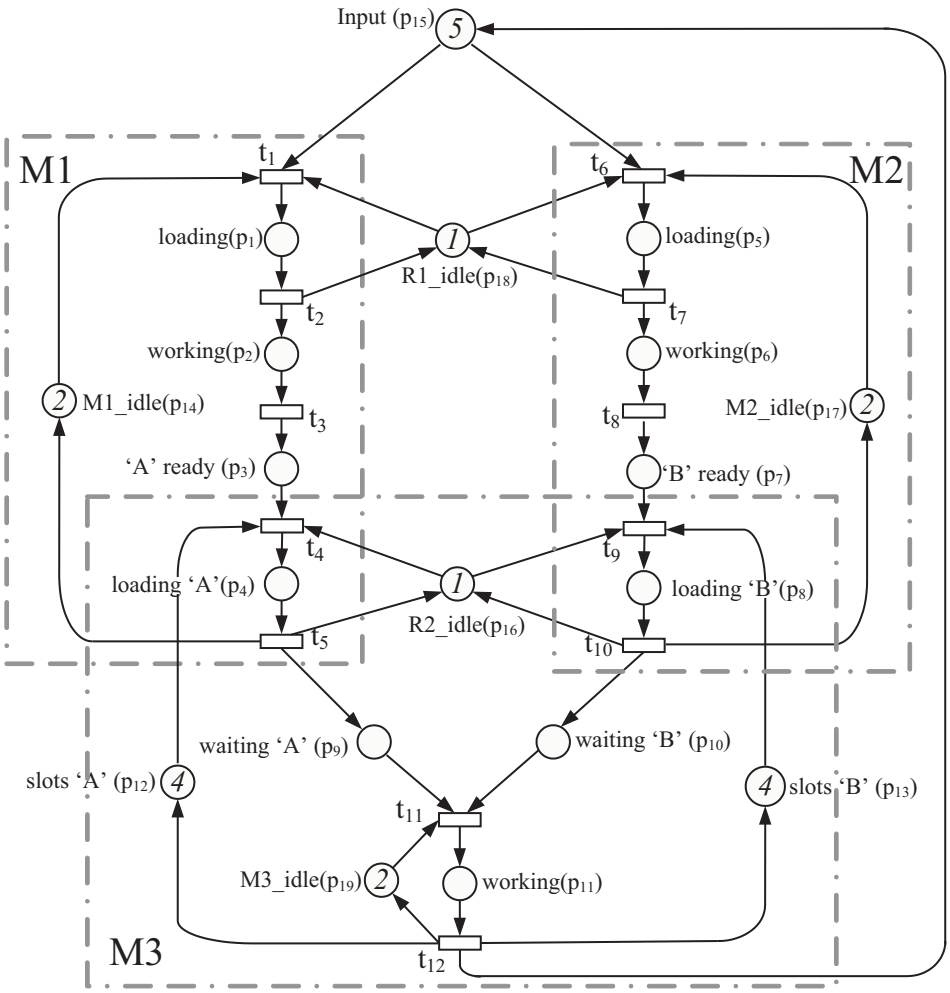


Figure 8.6: A variant of the PN model in Fig. 8.5: Robots R1, R2 are used in a different manner. Now R1 is used to move raw materials to M1 and M2, and R2 is used to move semi-products from M1 or M2 to M3.

The system is not CF (also the following other examples), so the standard ON/OFF controller is not applicable. Table A.5 (corresponding to the net system in Fig. 8.5) and Table A.6 (corresponding to the variant net system in Fig. 8.6) in the Appendix give the number of time steps required for reaching $\boldsymbol{m}_f$ by using different control methods, and the corresponding CPU time for computing the control laws.

For the original system in Fig. 8.5, the B-ON/OFF controller gives the smallest number of time steps (209), obtained by using $d = 2$. The ON/OFF+ controller

requires the largest number of time steps (457), although its computational cost (275ms) is only about half of the B-ON/OFF controller. The result of the approaching minimum-time controller is not very good, it requires 311 time steps to reach the final state and the required CPU time to compute the control law is 200 times as large as that of the B-ON/OFF controller. The number of time steps of the MPC-ON/OFF controller (around 220) is not far from the best, however its computational cost is high, for example, when $N = 5$ it is more than 200 times higher than that of the B-ON/OFF controller.

For the variant system in Fig. 8.6, the B-ON/OFF controller still gives the smallest number of time steps (223). The same number is also obtained by using the ON/OFF+ controller, while its computational cost is also the lowest (108ms, about one fourth of that of the B-ON/OFF controller). The approaching minimum-time controller requires the largest number of time steps (316) and very high computational cost (55,608ms). The MPC-ON/OFF controller also gives quite small number of time steps (228) but requires much more CPU time (even with $N = 1$ it is about 40 times larger than that of the ON/OFF+ controller.)

8.3 Case study 2: centralized control of an assembly line

The second example (adapted from [111]) is a more complex assembly line with five machines, three different parts A, B and C are assembled for one final product. The input of part A is first processed in machine M1 then machine M3; the input of part B is processed by machine M2 then machine M1; the semi-products from part A and B are first processed in machine M4; finally the obtained products are assembly in machine M5 with the one from part C that is sequentially processed by machine M3, M1 and M4. The production process of the system is shown in Fig. 8.7.

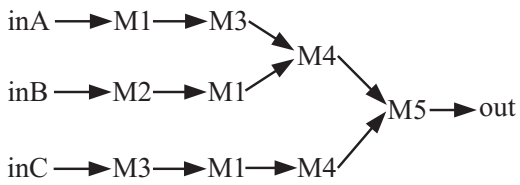


Figure 8.7: The production process of an assembly system

The PN model of the system is presented in Fig. 8.8. Machines are modelled by resource places labelled with name M1 to M5; each machine has buffers to store the semi-products, for instance, after the input of part A has been processed by M1, the obtained products are stored in buffer place $B1A(p_3)$, and the maximal sizes of the buffers are limited by, for example, place $MaxB1A(p_{23})$. The customer orders for the final products are also modelled and a "push" strategy is applied, i.e., the assembling process keeps working until the buffers are full. This strategy

can decrease the customer waiting time, accepting a high work in process. The interpretations of the places and transitions in the PN model are explained in Table 8.2.

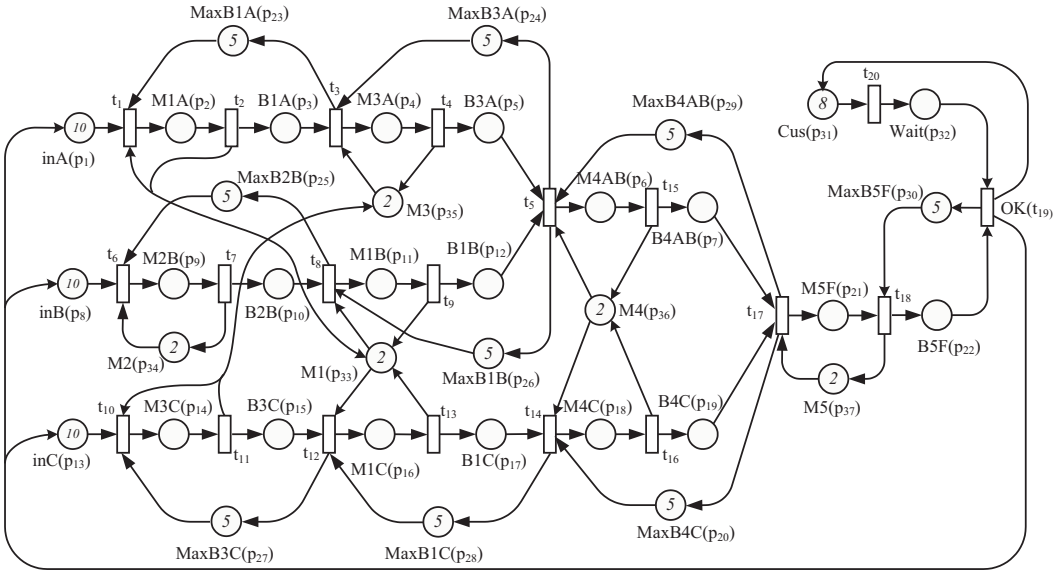


Figure 8.8: The PN model of an assembly system with five machines

Table 8.2: The interpretation of the PN model in Fig. 8.8

Place	Interpretation	Transition	Interpretation
inX	input of part X	t_1, t_8, t_{12}	M1 starts working
My	machine y	t_2, t_9, t_{13}	M1 finises
MyX	machine y is working on part X	t_6	M2 starts working
M5F	machine 5 is working on the final product	t_7	M2 finishes
ByX	buffer of part X in My	t_3, t_{10}	M3 starts working
B5F	buffer of the final product in M5	t_4, t_{11}	M3 finishes
MaxByX	maximal size of buffer ByX	t_5, t_{14}	M4 starts working
MaxB5F	maximal size of buffer B5F	t_{15}, t_{16}	M4 finishes
Cus	customers	t_{17}	M5 starts working
Wait	waiting orders	t_{18}	M3 finishes
		t_{19}	final product delivers
		t_{20}	customer order arrives

* X = A, B, C and y = 1, 2, 3, 4.

We assume that the input of each part initially has 10 pieces; each machine can handle 2 pieces at the same time; buffer sizes are limited to 5; and initially 8 customers are considered. The corresponding initial marking $\boldsymbol{m}_0$ is shown in Fig. 8.8, in which the empty places are assumed to have initial markings equal to 0.1. Similarly to the first example, we also assume that every transition has an average delay time, denoted by $\boldsymbol{\delta}$: the stating time delays of all machines are equal to 0.02 time units, i.e., $\boldsymbol{\delta}[t_1] = \boldsymbol{\delta}[t_3] = \boldsymbol{\delta}[t_5] = \boldsymbol{\delta}[t_6] = \boldsymbol{\delta}[t_8] = \boldsymbol{\delta}[t_{10}] = \boldsymbol{\delta}[t_{12}] = \boldsymbol{\delta}[t_{14}] = \boldsymbol{\delta}[t_{17}] = 0.02$; the working process of machine M1 takes 0.2 time units, i.e., $\boldsymbol{\delta}[t_2] = \boldsymbol{\delta}[t_9] = \boldsymbol{\delta}[t_{13}] = 0.2$; for machines M3 and M4, 0.4 time units, i.e., $\boldsymbol{\delta}[t_4] = \boldsymbol{\delta}[t_{11}] = \boldsymbol{\delta}[t_{15}] = \boldsymbol{\delta}[t_{16}] = 0.4$; machines M2, M5 work slower, it takes 0.8 time units, i.e., $\boldsymbol{\delta}[t_{7]} = \boldsymbol{\delta}[t_{18}] = 0.8$; the delay time of customer orders is 0.8; and the final products are delivered with delay of 0.1 time units, i.e., $\boldsymbol{\delta}[t_{19}] = 0.1$.

We consider reaching a final state with the flow of transition t_{19} (modelling the delivering of final products) maximized, and now we will also consider the work in process (WIP) cost in the profit function. We assume a fixed residence cost, equal to 50, for per piece of (semi-) products in the buffers; for machines M1, M4, the operation cost is 150; for machines M2, M5, the cost is 100; and for machine M3, the cost is 120. For per piece of final products, we assume an income of 1000. Then the following LPP can be written:

$$\begin{array}{ll} \max & J = 1000 \cdot \boldsymbol{w}_f[t_{19}] - \boldsymbol{l} \cdot \boldsymbol{m}_f \\ \text{s.t.} & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_f = \boldsymbol{0} \\ & \boldsymbol{w}_f[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^\bullet t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_f, \boldsymbol{\sigma}, \boldsymbol{m}_f \geq \boldsymbol{0} \end{array} \tag{8.2}$$

where J is the profit function and $\boldsymbol{l}[3] = \boldsymbol{l}[5] = \boldsymbol{l}[10] = \boldsymbol{l}[12] = \boldsymbol{l}[15] = \boldsymbol{l}[17] = \boldsymbol{l}[7] = \boldsymbol{l}[19] = \boldsymbol{l}[22] = 50$, $\boldsymbol{l}[2] = \boldsymbol{l}[11] = \boldsymbol{l}[16] = \boldsymbol{l}[6] = \boldsymbol{l}[18] = 150$, $\boldsymbol{l}[4] = \boldsymbol{l}[14] = 120$, $\boldsymbol{l}[9] = \boldsymbol{l}[21] = 100$, for other places p_i, $\boldsymbol{l}[i] = 0$.

By solving LPP (8.2), a desired final state is $\boldsymbol{m}_f = [5.78\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 1.024\ 0.05122\ 4.756\ 2.049\ 0.05122\ 0.5122\ 0.05122\ 5.78\ 1.024\ 0.05122\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 4.124\ 2.049\ 0.2561\ 4.637\ 4.124\ 3.1\ 4.637\ 4.124\ 4.637\ 4.124\ 4.844\ 7.844\ 0.2561\ 0.7634\ 0.05122\ 0.1512\ 0.1512\ 0.05122]^T$, with maximal flow $\boldsymbol{w}_f[t_{19}] = 2.561$. Table A.7 shows the simulation results by using different control methods.

In this case, the B-ON/OFF controller again gives the smallest number of time steps (131), which is much smaller than that of the ON/OFF+ controller (249) and that of the approaching minimum-time controller (300). By using the MPC-ON/OFF controller with $N = 5$, the same number of time steps as that of the B-ON/OFF controller can be obtained, but its computational cost (284,859ms) is about 600 times larger than that of the B-ON/OFF.

8.4 Case study 3: centralized control of a manufacturing system

In the third example (taken from [60]), we consider a flexible manufacturing system with four types of machines M1 to M4, and three type of robots R1 to R3. Three type of products P1 to P3 can be produced in this system. Fig. 8.9 represents its production process. The PN model of this system is shown in Fig.8.10, and the final products are finished in transition t_{14}, t_{20}, t_6 respectively.

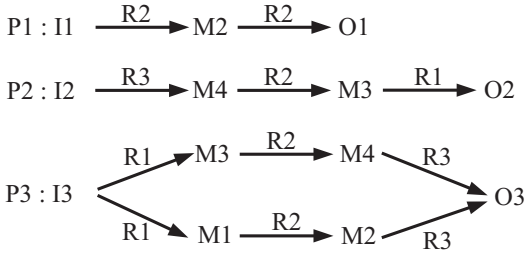


Figure 8.9: The production process of a flexible manufacturing system

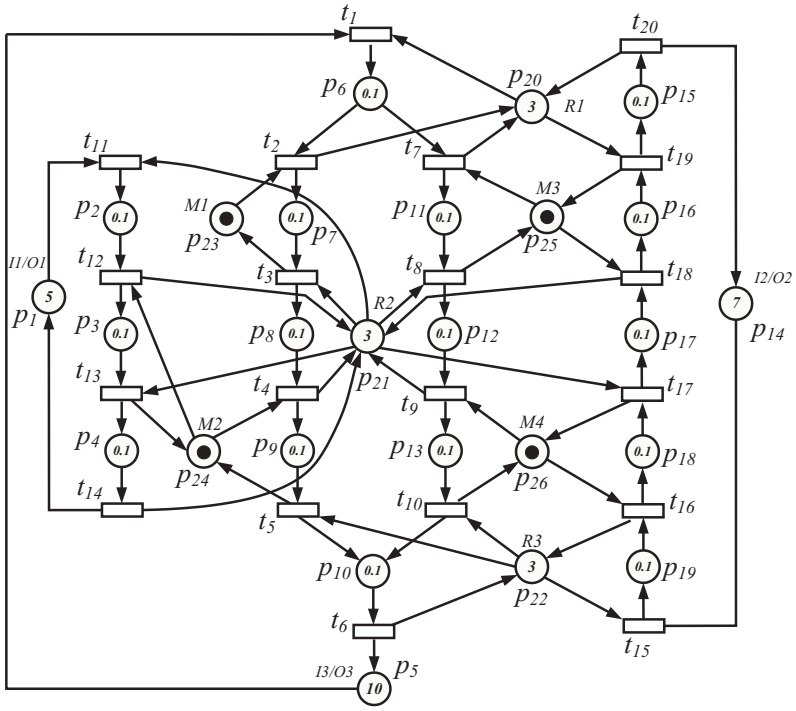


Figure 8.10: The PN model of a flexible manufacturing system

Assume that the initial state $\boldsymbol{m}_0(p_1) = 5$, $\boldsymbol{m}_0(p_5) = 10$, $\boldsymbol{m}_0(p_{14}) = 7$, $\boldsymbol{m}_0(p_{20}) =$

$\boldsymbol{m}_0(p_{21}) = \boldsymbol{m}_0(p_{22}) = 3$, $\boldsymbol{m}_0(p_{23}) = \boldsymbol{m}_0(p_{24}) = \boldsymbol{m}_0(p_{25}) = \boldsymbol{m}_0(p_{26}) = 1$ and the initial marking of the rest of places is equal to 0.1, the firing rate of transitions is defined as $\boldsymbol{\lambda}$ =[1/3 1/2 1/2 1/2 1/2 1 1/2 1/2 1/2 1/2 1/4 1/2 1/2 1 1/2 1/2 1/2 1/2 1/2 1/4]^T and sampling period be $\Theta = 0.03$. Let us assume that the objective function of the system is defined as: $3 \cdot \boldsymbol{w}_f[t_6] + 5 \cdot \boldsymbol{w}_f[t_{14}] + 4 \cdot \boldsymbol{w}_f[t_{20}]$, where $\boldsymbol{w}_f$ is the flow in a steady state. Then following LPP can be written:

$$\begin{array}{ll} \max & J = 3 \cdot \boldsymbol{w}_f[t_6] + 5 \cdot \boldsymbol{w}_f[t_{14}] + 4 \cdot \boldsymbol{w}_f[t_{20}] \\ \text{s.t.} & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_f = \boldsymbol{0} \\ & \boldsymbol{w}_f[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^\bullet t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_f, \boldsymbol{\sigma}, \boldsymbol{m}_f \geq \boldsymbol{0} \end{array} \tag{8.3}$$

By solving LPP (8.3), a desired final state (maximizing the profit function) can be computed: $\boldsymbol{m}_f$ = [3.4; 1.3; 0.4; 0.2; 7.6; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 3.1; 0.8; 0.4; 0.4; 0.4; 2.4; 2; 0.8; 0.4; 0.7; 0.4; 0.4; 0.4] and the maximal profit is $J = 3$. The simulation results by using different control methods are shown in Table A.8.

In this example, the approaching minimum-time controller gives the smallest number of time steps (279), which is about 10% smaller than that of the other control methods. However, the CPU time required for computing the control law (60,691ms) is about 200 times as large as that of the ON/OFF+ controller and 50 times as large as that of the B-ON/OFF controller.

8.5 Discussions of case study 1–3

In Table 8.3, we summarize the smallest numbers of time steps that obtained by using different control methods, and the corresponding parameters, according to the simulation results shown in Table A.5–A.8. From the simulation results, we can make some initial conclusions (that are consistent with those in Section 4.4.5):

Regarding the numbers of time steps:

- The B-ON/OFF controller gives the smallest number of time steps in most of the cases (except for case study 3). Usually smaller numbers of time steps are obtained by using smaller values of d, as shown in Table A.5 and Table A.7. A small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows, therefore a smaller value of d may give better results. If d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to apply the ON/OFF+ directly. However, in the case that the B-ON/OFF controller cannot improve the result of the ON/OFF+ controller

Table 8.3: The smallest numbers of time steps of using different control methods among different parameters (if exist), derived from Table A.5–A.8. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Cases	Control methods	Time steps	CPU time	Parameters
Case 1	appro. min-time	311	88,505	
	ON/OFF+	457	275	
	B-ON/OFF	209	426	$d = 2$
	MPC-ON/OFF	219	102,622	$N = 5$, $r = 1000$, q =1 (or 100, 1000)
Case 1 (variant)	appro. min-time	316	55,608	
	ON/OFF+	223	108	
	B-ON/OFF	223	419	$d = 20$ (or 15, 10, 5, 2)
	MPC-ON/OFF	228	5,018	$N = 1$, r=1000, q=1
	MPC-ON/OFF	228	99,749	$N = 5$, $r = 1000$, q =1 (or 100, 1000)
Case 2	appro. min-time	300	138,826	
	ON/OFF+	249	213	
	B-ON/OFF	131	475	$d = 10$ (or 5, 2, 1)
	MPC-ON/OFF	131	284,859	$N = 5$, r=1000, q=100
Case 3	appro. min-time	279	60,691	
	ON/OFF+	301	271	
	B-ON/OFF	301	1,157	$d = 20$ (or 15, 10, 5, 2, 1)
	MPC-ON/OFF	310	443,439	$N = 5$, $r = 1000$, q =1 (or 100, 1000)

(when conflicting transitions have similar flows), the numbers of time steps are not sensitive to the values of d, as shown in Table A.6 and Table A.8.

- The ON/OFF+ controller usually gives quite small numbers of time steps (as shown in Table A.6 and Table A.8), except when the flows of conflicting transitions are very different. For example, in the original system in case study 1 (Fig. 8.5, Table A.5), the flows of t_4 and t_9 are much smaller than the ones of t_1 and t_6; those four transitions are in a coupled conflict, therefore if we fire them proportionally by using the ON/OFF+ strategy the result is not good, costing 457 time steps, more than the double of the B-ON/OFF controller, to reach $\boldsymbol{m}_f$. On the other hand, we can see that in the variant system in cast study 1 (Fig. 8.6, Table A.6), it gives the smallest number of time steps (as the B-ON/OFF controller) and its computational cost is the lowest. It is because the fact that in the variant system, transitions t_4, t_9 and transitions t_1, t_6 are now in different sets of coupled conflict.

- Most probably, the approaching minimum-time controller may not work very well when there exist some places with very small initial markings (as in case study 1 and 2, Table A.5–A.7). The reason may be what have mentioned before: in this approach the flow between two adjacent states is determined

by the one with the smaller flow, so if some states have very small flows (for example the initial state), it may take long time to reach the final state.

- The numbers of time steps obtained by using the MPC-ON/OFF controller are not far from the best among other control methods, even the best (as the B-ON/OFF controller) in case 2 (shown in Table A.7). Usually, its required numbers of time steps can be decreased but using a larger time horizon N, but the improvement is not very significant (less than 0.5% in case 1, 3% in case 3 and 10% in case 2, but the computational costs increase more than 20 times). However, we can also observe that even with a small N, we may already obtain a reasonable number of time steps (as in case 1, Table A.5 and Table A.6). On the other hands, in these examples the number of time steps is not very sensitive to the weights on the matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Regarding the computational costs:

- The ON/OFF+ controller is the computationally cheapest one in all these three case studies.
- The B-ON/OFF controller costs slightly higher CPU time than the ON/OFF+ controller (less than 5 times higher), because an estimation of number of time steps should be computed at each time step. But, it is still very efficient.
- We can see that in those three cases, the approaching minimum-time controller costs more than hundred times the CPU time than that of the ON/OFF+ and B-ON/OFF controller, because it needs to solve a non-linear problem whenever an intermediate state is inserted on the trajectory to reduce the time.
- In these examples, the computational cost of MPC-ON/OFF controller is not higher than the approaching minimum-time controller when a small N is used; but it always costs more CPU time than the ON/OFF+ and B-ON/OFF controller. On the other hand, the computational cost grows fast with respect to N. From our simulation, the required CPU time for computing the control law increases more than 20 times, when N increases from 1 to 5. (We should also notice that, the increasing speed of the computational costs with respect to N may also depend on the size and structure of the considered net systems.)

8.6 Case study 4: distributed control of an AGV System

In this case study we consider a model of Automatic Guided Vehicle Systems(AGVS). The system describes a manufacturing factory floor that consists of four workstations: WS 1 to 4 and three AGVS areas: AGVS 1 to 3. Each workstation handles some parts of the system that are first stored in the buffers, then moved from one workstation to another through the AGVS areas. The system can be naturally viewed

as 7 subsystem connected by those buffers, each workstation or AGVS area as a subsystem. The PN model of the system is presented in Fig. 8.11, in which the buffers are modelled as places. The input and output buffer places of each subsystem are: for subsystem S^1, $B^{(\cdot,1)} = \{p_1\}$, $B^{(1,\cdot)} = \{p_9, p_{10}\}$; for subsystem S^2, $B^{(\cdot,2)} = \{p_9, p_{10}\}$, $B^{(2,\cdot)} = \{p_{16}, p_{24}\}$; for subsystem S^3, $B^{(\cdot,3)} = \{p_{16}\}$, $B^{(3,\cdot)} = \{p_{30}\}$; for subsystem S^4, $B^{(\cdot,4)} = \{p_{24}\}$, $B^{(4,\cdot)} = \{p_{27}\}$; for subsystem S^5, $B^{(\cdot,5)} = \{p_{30}, p_{27}\}$, $B^{(5,\cdot)} = \{p_{33}, p_{36}\}$; for subsystem S^6, $B^{(\cdot,6)} = \{p_{33}, p_{36}\}$, $B^{(6,\cdot)} = \{p_{45}\}$; for subsystem S^7, $B^{(\cdot,7)} = \{p_{45}\}$, $B^{(7,\cdot)} = \{p_1\}$.

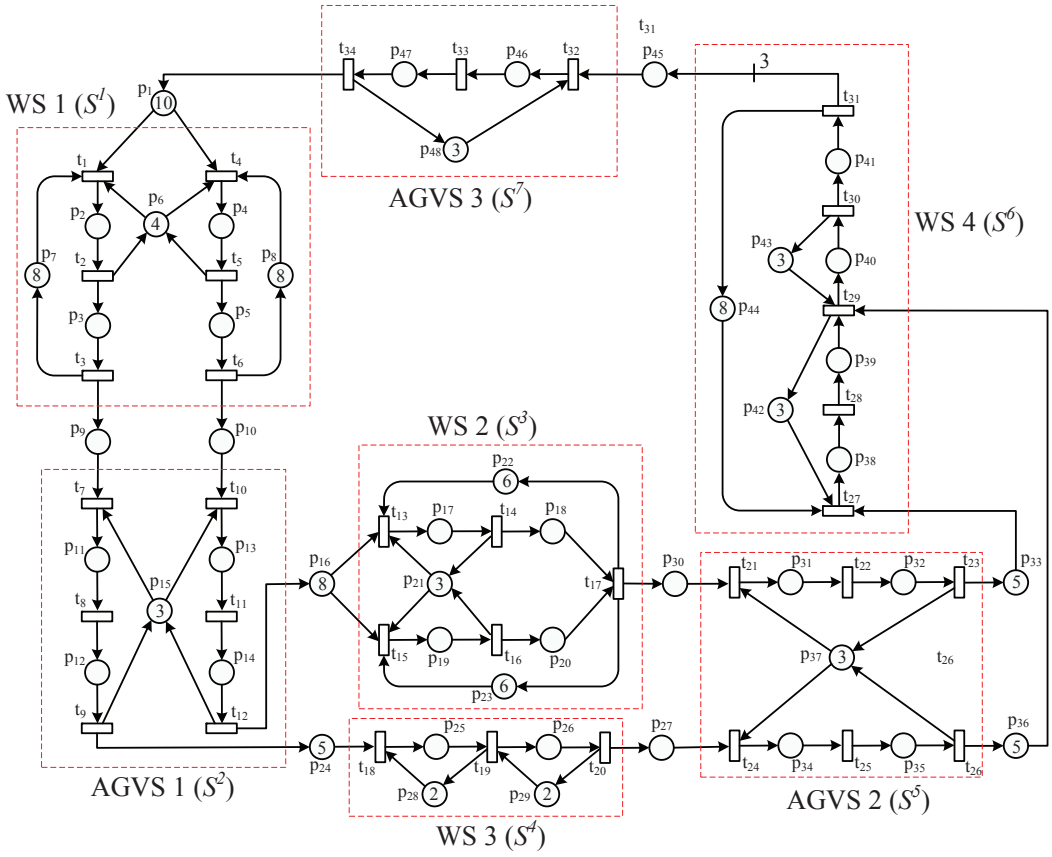


Figure 8.11: The PN model of an AGVS composed by 7 subsystems

We assume that in the initial state there are certain raw materials or parts in the input buffer places of each workstation, in particular, $\boldsymbol{m}_0[p_1] = 10$; $\boldsymbol{m}_0[p_{16}] = 8$; $\boldsymbol{m}_0[p_{24}] = 5$; $\boldsymbol{m}_0[p_{33}] = \boldsymbol{m}_0[p_{36}] = 5$. In each workstation there exist some (shared) machines, modelled by places, which have the initial markings as the following: $\boldsymbol{m}_0[p_6] = 4$; $\boldsymbol{m}_0[p_{21}] = 3$; $\boldsymbol{m}_0[p_{28}] = \boldsymbol{m}_0[p_{29}] = 2$; $\boldsymbol{m}_0[p_{42}] = \boldsymbol{m}_0[p_{43}] = 3$. There also exist limitations for the parts that can be accepted by the workstations: $\boldsymbol{m}_0[p_7] = \boldsymbol{m}_0[p_8] = 8$; $\boldsymbol{m}_0[p_{22}] = \boldsymbol{m}_0[p_{23}] = 6$; $\boldsymbol{m}_0[p_{44}] = 8$. We assume that in each AGVS area, there maximally have 3 available vehicles, i.e.,

$\boldsymbol{m}_0[p_{15}] = \boldsymbol{m}_0[p_{37}] = \boldsymbol{m}_0[p_{48}] = 3$. For the other places in model, we assume their initial markings equal to 0.1 ensuring positiveness of the initial state required by the control method. Similarly to the previous examples, we assume final states of subsystems given in Table 8.4 such that the maximal flow may be obtained. The final states of buffer places are: $m_f[p_1] = 4.23$; $m_f[p_9] = 0.14$; $m_f[p_{10}] = 0.27$; $m_f[p_{16}] = 4.80$; $m_f[p_{24}] = 3.60$; $m_f[p_{27}] = 0.14$; $m_f[p_{30}] = 0.14$; $m_f[p_{33}] = 1.57$; $m_f[p_{36}] = 3.00$; $m_f[p_{45}] = 0.41$. We assume that the average delay times of transitions are set to the values shown in Table 8.5 and the sampling period $\Theta = 0.05$.

Table 8.4: The final states of subsystems

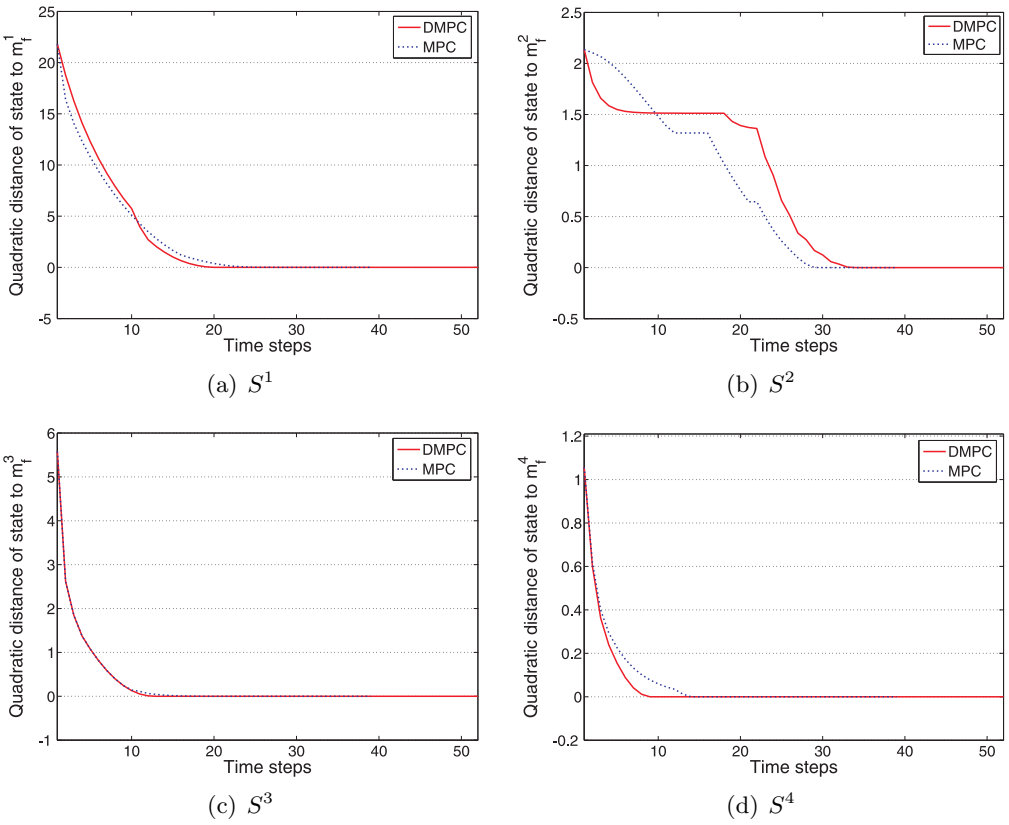
S^1		S^2		S^3		S^4		S^5		S^6		S^7	
p_2	0.68	p_{11}	0.40	p_{17}	0.54	p_{25}	0.68	p_{31}	0.40	p_{38}	1.08	p_{46}	1.22
p_3	0.54	p_{12}	0.14	p_{18}	0.81	p_{26}	0.54	p_{32}	0.14	p_{39}	0.54	p_{47}	0.40
p_4	1.36	p_{13}	0.81	p_{19}	0.68	p_{28}	1.42	p_{34}	0.40	p_{40}	1.08	p_{48}	1.57
p_5	2.17	p_{14}	0.27	p_{20}	0.81	p_{29}	1.56	p_{35}	0.14	p_{41}	0.54		
p_6	2.17			p_{21}	1.98			p_{37}	2.32	p_{42}	1.57		
p_7	6.98			p_{22}	4.85					p_{43}	2.02		
p_8	4.68			p_{23}	4.71					p_{44}	5.15		

Table 8.5: The average delay times of transitions

S^1		S^2		S^3		S^4		S^5		S^6		S^7	
t_1	0.4	t_7	0.1	t_{13}	0.3	t_{18}	0.4	t_{21}	0.1	t_{27}	0.4	t_{32}	0.1
t_2	0.5	t_8	0.3	t_{14}	0.4	t_{19}	0.5	t_{22}	0.3	t_{28}	0.8	t_{33}	0.3
t_3	0.4	t_9	0.1	t_{15}	0.5	t_{20}	0.4	t_{23}	0.1	t_{29}	0.4	t_{34}	0.1
t_4	0.8	t_{10}	0.1	t_{16}	0.5			t_{24}	0.1	t_{30}	0.8		
t_5	0.5	t_{11}	0.3	t_{17}	0.6			t_{25}	0.3	t_{31}	0.4		
t_6	0.8	t_{12}	0.1					t_{26}	0.1				

The problem we handle here is to drive all the subsystems to their final states by using distributed control methods. Clearly, the net is not CF, so the approach proposed in Chapter 5 is not applicable; on the other hand, subsystems are not mono-T-semiflow (for example, subsystem S^1 has two minimal T-semiflows), so we cannot apply the method proposed in [4] either. Hence, here we will apply the distributed MPC controller presented in Chapter 6. In Fig. 8.12 and Fig. 8.13, for each subsystem S^l we show the *quadratic distance* of states $\boldsymbol{m}_k^l$ (at any time step k) to the final state $\boldsymbol{m}_f{}^l$, defined as $(\boldsymbol{m}_k^l - \boldsymbol{m}_f{}^l)^T \cdot (\boldsymbol{m}_k^l - \boldsymbol{m}_f{}^l)$, $l = 1$ to 7, by using the centralized MPC controller given in Algorithm 8 and the distributed MPC controller given in Algorithm 9. Since subsystems may not reach their final states at the same time, Fig. 8.13(d) shows the different time instants of reaching the final states by using the centralized and distributed MPC. The results are obtained by

using $N = 3$, $\boldsymbol{Q} = \boldsymbol{I}$, $\boldsymbol{Z} = 1000 \cdot \boldsymbol{I}$. For other parameters in distributed MPC, we use $\alpha = 0.5$, $\epsilon_1 = 0.1$, $\epsilon_2 = 10^{-6}$.

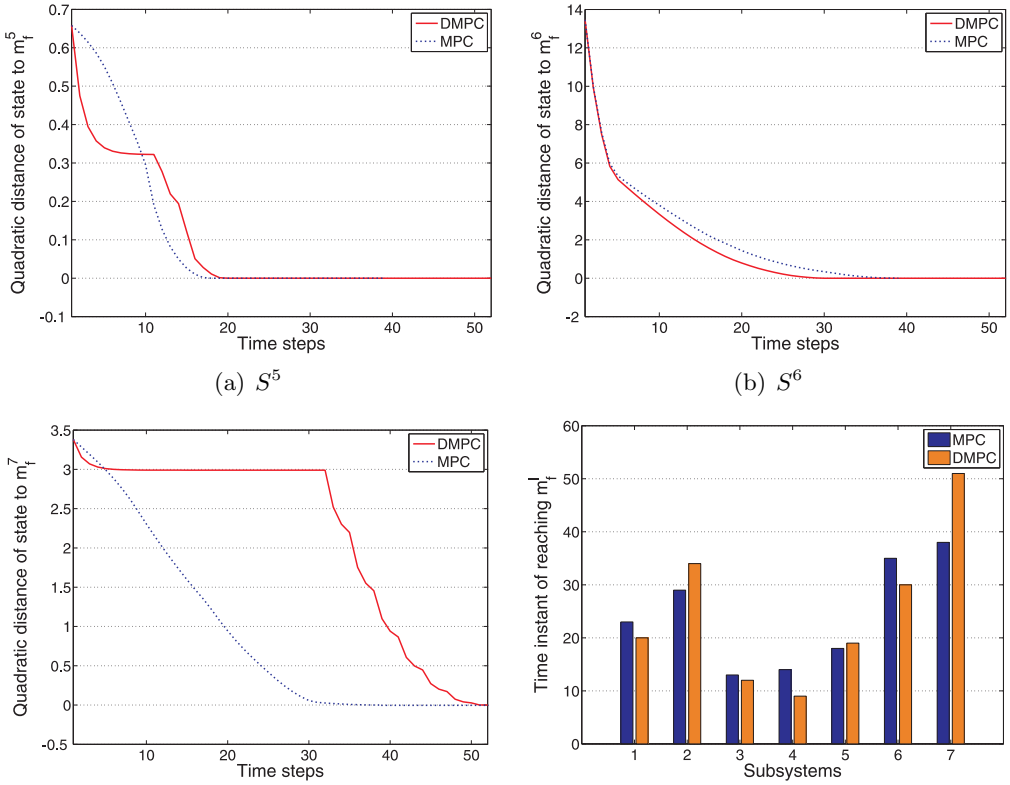


(a) S^1 (b) S^2

(c) S^3 (d) S^4

Figure 8.12: The quadratic distance of $\boldsymbol{m}_k^l$ to the final state ${\boldsymbol{m}_f}^l$, $l = 1, 2, 3, 4$

It can be observed that all the subsystems reach their final states in finite time by using both the centralized and distributed methods. From Fig. 8.13(d) we can see that, the last subsystem that reaches its final state is S^7, requiring 38 time steps by using the centralized MPC and 51 time steps by using the distributed MPC. However, let us notice that not all the subsystems reach their final states slower by using the distributed MPC. For instance, the distributed MPC controller requires 9 time steps for subsystem S^4 to reach ${\boldsymbol{m}_f}^4$, smaller than the 14 time steps of using the centralized MPC. On the other hand, Fig. 8.15(a) shows that by using the distributed MPC, the CPU time consumed for computing the control laws (the sum of the CPU time consumed in all the subsystems) is much smaller than by using the centralized MPC, and the reason is clear: the computational complexity of MPC based approaches may grow very quickly on the size of net systems. We should point out that, in both centralized and distributed MPC approaches, from each time step k the state evolutions of subsystems are constrained to be inside the subset convex $R(\mathcal{N}, \boldsymbol{m}_k, \boldsymbol{m}_f)$ (generated by constrains (6.2d) and (6.2e)) of the reachability space.



(a) S^5 (b) S^6

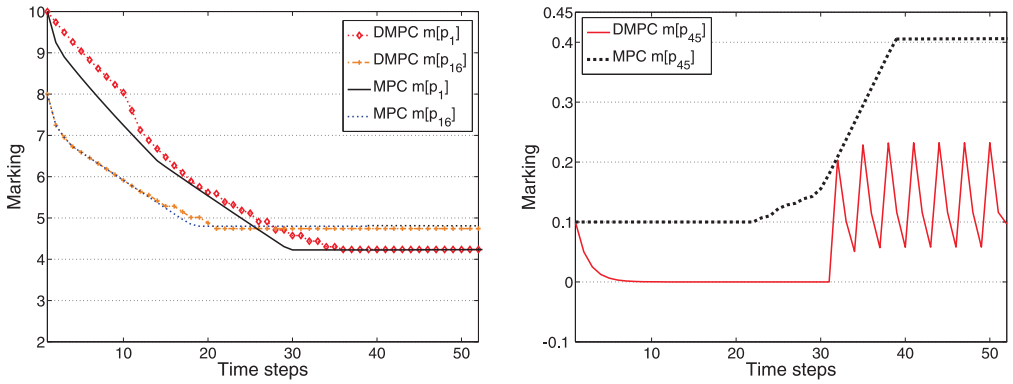
(c) S^7 (d) Time instants

Figure 8.13: (a)-(c) the quadratic distance of $\boldsymbol{m}_k^l$ to the final state ${\boldsymbol{m}_f}^l$, $l = 5, 6, 7$; (d) the time instants of subsystems reaching to ${\boldsymbol{m}_f}^l$, $l = 1$ to 7

Therefore we can see that the states converge to the final ones monotonically.

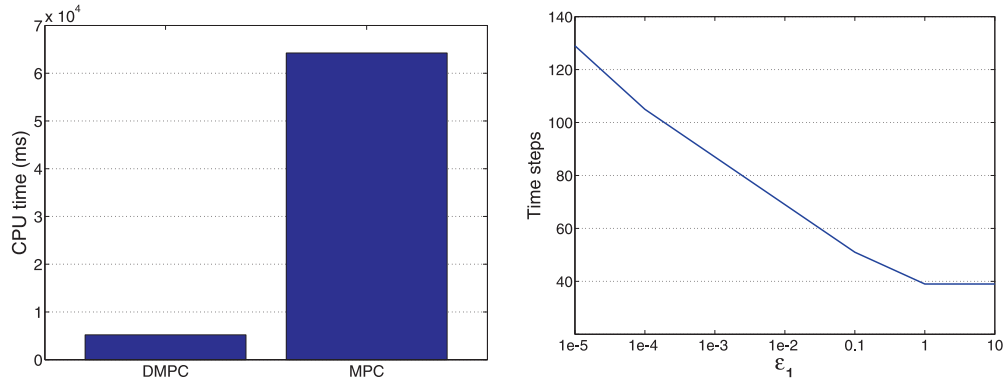
However, in the distributed method the markings of buffers are not controlled as in the centralized method, thus have more "freedom". Therefore we may not be able to conclude which one is better in general, just considering the time spent on the trajectories. Fig. 8.14 shows the state evolutions of buffer places (we take p_1, p_{16} and p_{45} as examples). We can observe that the markings of buffer places may reach different values: by using the centralized MPC the final markings specified in $\boldsymbol{m}_f$ are reached; but by using the distributed MPC, the markings of buffer places may reach some different (non-negative) values since here we only consider the convergence to the final states of subsystems. By using the distributed MPC the marking trajectory of buffer places, e.g., p_{45} in Fig. 8.14(b), is no longer monotone. For instance, after 31 time steps, the marking of p_{45} oscillates around the value of 0.1. This is because subsystem S^6 reaches its final state at 30 time steps (see Fig. 8.13(d)), then optimization problem (6.11) should be solved in S^6 and it tries to put more tokens to its (unique) output buffer place p_{45} when its marking $\boldsymbol{m}_k[p_{45}] \leq \epsilon_1 = 0.1$ (see cost function (6.9) and Remark 6.3.1); at the same time, subsystem S^7 needs to consume

tokens from its input buffer place p_{45} for evolving towards its final state.



(a) The state evolution of p_1 and p_{16} (b) The state evolution of p_{45}

Figure 8.14: The state evolution of some buffer places



(a) (b)

Figure 8.15: (a) The CPU time of using centralized and distributed MPC. (b) The time steps of using the distributed MPC with different ϵ_1

By using the distributed MPC controller given in Algorithm 9, each subsystem S^l tries to put more tokens to its output buffer places $p_i \in B^{(l,\cdot)}$ (by solving the optimization problem (6.11)) only if it is not able to evolve towards its final state (or its final state has already been reached), and $\boldsymbol{m}_k[p_i] \leq \epsilon_1$ where ϵ_1 is assumed to be very small positive value to approximate $\boldsymbol{m}_k[p_i] = 0$ (in cost function (6.9)). One may think that when problem (6.11) is solved, if we could use a larger value for ϵ_1, i.e., if we could also try to put tokens into those output buffer places of S^l that are not "nearly emptied", to make its neighboring subsystems evolve faster. However, in this case the convergence to final states may not be guaranteed when two neighboring subystems are both solving problem (6.11) and there exist multiple input/output buffers between them. In this particular example, subsystems can

converge to their final states faster if a larger value for ϵ_1 is used, shown in Fig. 8.15(b). Notice that when $\epsilon_1 \geq 1$, its number of time steps required to reach final states (39 time steps) is almost the same as using the centralized MPC (38 time steps).

8.7 Conclusions

In this chapter, we carry out several case studies to illustrate the control methods proposed in this thesis for the target marking control problem of TCPNs under infinite server semantics.

In the first three case studies, we focus on the centralized ON/OFF based control methods, aiming at driving the system to the final state and minimizing the time spent on state evolution. It is shown that an advantage of the ON/OFF based controllers is the low computational complexity: in all the case studies (simulations results shown in Table A.1–A.8), the ON/OFF+ and B-ON/OFF controllers have lower computational costs than the approaching minimum-time controller proposed in [5]; while for the MPC-ON/OFF controller with a small time horizon N, its computational cost may be also not higher (as for the systems in case studies 1–3). At the same time, reasonable numbers of time steps for reaching the final state can be obtained. The standard ON/OFF controller is the most suitable choice for a CF net system, ensuring low computational complexity and minimum-time. For non-CF nets, some characteristics of the system may be helpful in choosing an appropriate method. In particular, if the flows (depending on markings) of conflicting transitions are very different, the B-ON/OFF controller may obtain a smaller number of time steps than the ON/OFF+ controller. The approaching minimum-time controller [5] may not be a good choice if there are some places with very small markings (because in this approach, between each pair of adjacent states of the trajectory the firing speed is constant and determined by the one with a smaller flow); in those cases the ON/OFF based methods usually can achieve better results. It may be interesting to point out that in "slow" practical systems like logistics or manufacturing systems, the operational time may be much larger than the computational time for the control laws (considering the very low computational complexity of ON/OFF based methods; including the MPC-ON/OFF controller with small time horizon); therefore, at each operation instant we could compute the control laws by using several of the methods, and then choose the best one to apply.

In the last case study, we apply the distributed MPC controller proposed in Chapter 6. The other decentralized control methods proposed in Chapter 5 and contribution [4] are not applicable to this example, because the net or some subnets are not CF or mono-T-semiflow. We show that by using the distributed MPC controller given by Algorithm 9, all the subsystems reach their final state in finite time (but not minimum-time in general), while all the buffer places shared by subsystems are always in legal non-negative states.

Chapter 9

Conclusions and Future works

Many man-made systems, such as manufacturing, logistics, telecommunication or traffic systems, can be "naturally" viewed as Discrete Event Dynamic Systems (DEDS). Nevertheless, large populations or heavy traffic frequently appear and they may suffer from the classical *state explosion* problem. In order to overcome this problem, *fluidization* can be applied, obtaining the fluid relaxation of the original discrete model. Continuous Petri nets (CPNs) are a fluid approximation of discrete Petri nets (PNs), a well known formalism for DEDS. It is obtained by removing the integrality constraints on the firings of transitions, thus on the markings (states). One key benefit of using CPNs is that, most frequently, it leads to a substantial reduction in the computational costs. For instance, several techniques based on integer programming in the discrete model may be solved using linear programming in the fluid case. Among other interesting issues, many works can be found in the literature considering the modelling, analysis, observability and control of CPNs.

In this thesis we have focused on the control of timed continuous Petri nets (TCPNs), in which time interpretations are associated to transitions. Depending on how the flow of transitions is defined, there exist different server semantics and we assume the *infinite server semantics* (variable speed) because it has been observed to usually obtain better approximations of discrete PNs in general, and always under some particular conditions (net subclasses).

Regarding the control problem, the first question is what to control? We assume that control actions are applied to *slow down* the firing of transitions [89], i.e., to reduce the flow. We have considered two interesting control problems in this thesis:

- The first problem is called *target marking control*, where the objective is to drive the system (as fast as possible) from an initial state $\boldsymbol{m}_0$ to a desired final state $\boldsymbol{m}_f$. It is similar to the *set-point* control problem in a general continuous-state system. By considering the CPNs as a relaxation of discrete models, the continuous state can be viewed as an approximation of the average state in the original discrete system.

- The second problem is called *optimal flow control*, in which the objective is to drive the system to an optimal flow (obtained in a "convex final region"),

without *a priori* knowledge of a specific final state. Usually, these final states with the optimal flow are not unique and they belong to a convex region. In this work, we are interested in reaching as fast as possible the maximal flow, what is frequently desirable in practical systems.

In both control problems, we address the minimum-time evolution, a problem that is close to the minimization of the *makespan* in manufacturing (time difference between the start and finish of a sequence of jobs or tasks). In many existing control methods, the computational costs may grow very quickly (for example, the approaching minimum-time controller [5], because computationally expensive BPPs should be solved), even exponentially (for example, the affine control [102], because the number of vertices increases exponentially on the number of places), with respect to the size of the net system. In this work we use an ON/OFF strategy, which is shown to be very efficient. We consider both centralized and decentralized settings for the target marking control problem.

Regarding the *centralized* methods we have developed several (heuristic) minimum-time controllers based on the ON/OFF strategy, which is frequently used in minimum-time problems:

- **ON/OFF controller**: This controller is specially suitable for Choice-Free (CF) nets. A interesting property of CF nets is that they are structurally persistent, i.e., for any initial state ($\boldsymbol{m}_0$), the enabling degree of a transition will not be decreased by firing other ones; furthermore, the reaching of one (final) state does not depend on the firing orders (speeds) of transitions, and it is only relevant to the required firing count vector. We have shown that the desired final state can be reached in minimum-time by using a simple ON/OFF strategy: fire every transition as fast as possible (ON) until an upper bound, given by the *minimal firing count vector*, is reached; then simply block it (OFF).

- **ON/OFF+ controller**: In the case that the net is not CF, we may not be able to apply the ON/OFF controller, because its "greedy" strategy of firing *conflicting* transitions may bring some "blocked" situations, even for live and bounded net systems. The ON/OFF+ controller overcomes this problem by forcing proportional firings of conflicting transitions according to a given firing count vector; while for the persistent transitions, the standard ON/OFF strategy is applied.

- **B-ON/OFF controller**: The ON/OFF+ controller requires very low computation costs, but it may have a drawback in some cases where the flows of conflicting transitions are very different: if conflicting transitions fire proportionally, then obviously, the overall time spent for firing a given firing count vector to reach $\boldsymbol{m}_f$ is determined by the "slower" ones. In the B-ON/OFF controller, this problem is handled by adding a "balancing process", i.e., we first fire the "faster" transitions and block the "slower" transitions for a period of time, until they get *balanced* (if they can). After that, we simply apply the

ON/OFF+ controller. In this way, the time spent for reaching the final state may be decreased, comparing with that of the ON/OFF+ controller.

- **MPC-ON/OFF controller**: Model Predictive Control (MPC) has been widely applied in industry for process control. Usually it is used for optimizing trajectory by solving certain optimization problems at each time step. In this work, we have tried to combine the MPC scheme with the ON/OFF strategy, obtaining a heuristics for approaching minimum-time target marking control. In particular, conflicts are solved by using the MPC approach; while the ON/OFF strategy is applied to persistent transitions.

All the proposed ON/OFF-based controllers ensure the *convergence*. For CF nets, the standard ON/OFF controller gives a minimum-time evolution; while other controllers are heuristics for minimum-time target marking control of general net systems and, we have seen in examples that reasonably small numbers of time steps for reaching the final state can be obtained. A main advantage of using the ON/OFF strategy is its low computational complexity. In ON/OFF, ON/OFF+ and B-ON/OFF controllers, only simple linear programming problems need to be solved at each time step. The MPC-ON/OFF controller may be computationally more expensive (a QPP problem is involved at each time step) and the computational cost grows fast with respect to the time horizon N. For example, for the net system in Fig. 4.11 (simulation results shown in Table A.1–A.4) the required CPU time increases about 50 times when N increases from 1 to 10. Nevertheless, even if a smaller number of time steps may be obtained by using a larger N, from our simulations, in some cases the improvement is not very significant; and even a worse result may be obtained using a larger N (e.g., in the case of setting $\mathbf{s}.1$) for the net system in Fig. 4.11); so using a small N may be a reasonable choice.

Decentralized control methods become interesting for large scale and usually physically distributed systems, for which centralized control may be difficult to implement. The existing work related to the decentralized control of TCPNs is still very limited, in this thesis we have proposed two approaches:

- **Decentralized minimum-time control of CF nets**: We assume that large scale systems are cut into subsystems through sets of buffer places (intersections). The idea is that we first compute the local control laws (minimal firing count vectors) separately in all subsystems, then we reach an agreement among them to get globally admissible control laws. In order to ensure that identical behaviors (firing sequences) of the original system are obtained in subsystems, we have employed several reduction rules to build abstractions of the missing parts of disconnected subsystems; for reaching the agreement among local control laws, a high level coordinator is introduced. When globally admissible control laws are obtained, the ON/OFF controller can be implemented independently in any subsystem, obtaining a minimum-time evolution.

- **Distributed MPC control of general nets**: For general net systems, the decentralized control method proposed for CF nets may not be applicable and

one reason is that we may not be able to achieve an agreement among local control laws. We have proposed a distributed control method for general nets based on the MPC approach. First, we present a *centralized* MPC controller, in which the state evolution is constrained to be inside an interior convex subset of the reachability space and thus, the convergence to the final state can be guaranteed. This approach is less constrained than the methods proposed in [5] and [64], in which linear trajectories were considered. Later, we have proposed a distributed MPC control algorithm. Similarly to the previous method for CF nets, we assume that subsystems are connected by sets of *buffer* places. A key issue in the distributed setting concerns the strict positiveness of the markings of buffer places: an alternative optimization problem is considered to recover from the situations where the markings of some buffer places are converging to zero.

In the decentralized control method for CF nets, we construct the *complemented subsystems* by using several reduction rules and the identical behaviros of the original system are preserved. Let us point out that those obtained complemented subsystems may also be useful in other contexts, for example, throughput approximations in a distributed way (similar works have been done for discrete models, for example in [20] for marked graphs, and later in [75] for weighted T-systems). The distributed MPC control method proposed here is not designed for the minimum-time control, but it can be applied for general nets and ensures that all the subsystems converge to their final states. As a by-product, the MPC controller applied in each subsystem can also be used independently in centralized setting.

The (minimum-time) optimal flow control problem has been considered here for CF nets. Usually, we may obtain the optimal flow (in our case, the maximal flow) of the system in a set of steady states, belonging to a convex subset in the reachability space. Since we cannot uniquely determine a desired final state, even for MG (a subclass of CF nets) the minimum-time flow control becomes difficult. In this work, we have proposed a heuristic algorithm for CF nets, in which we compute the "best" firing count vector bringing the system to the maximal flow, according to an estimation of the number of time steps based on the current state and flow; and an ON/OFF firing strategy is applied. We also show that because of the persistency of CF nets, we can further reduce the time spent to reach the maximal flow by means of some additional firings.

Despite the many results that can be found in the literature about the control of TCPNs, and some new methods proposed in this thesis, there still remain many widely open issues that deserve more investigations. Among many others:

1) In the proposed methods, the ON/OFF strategy is applied to achieve minimum-time evolution. It could be also important to consider the minimum-time control that will satisfy additional conditions. For example, control laws that require minimal control energy could be interesting.

2) More extensive comparisons of the available control methods are necessary, providing more concrete criteria of selecting suitable methods in different situations. For this aim, certain benchmarks with systems of different net structures, markings and firing rates are desirable.

3) Furthermore, comparisons should be performed also using the discrete models. A control method that works well for the fluid model may not be a good choice when the control laws are interpreted and applied to control the underlying discrete one.

4) Most of the existing results, including this work, assume that all the transitions are controllable. A clear extension is to further consider partially controllable systems.

Bibliography

[1] H. Alla and R. David. Continuous and Hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8:159–188, 1998.

[2] E. Altman, T. Jiménez, and G. Koole. On the comparison of queueing systems with their fluid limits. *Probability in the Engineering and Informational Sciences*, 15(2):165–178, 2001.

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

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

[5] H. Apaydin-Ozkan, J. Júlvez, C. Mahulea, and M. Silva. Approaching Minimum Time Control of Timed Continuous Petri nets. *Nonlinear Analysis: Hybrid Systems*, 5(2):136–148, May 2011.

[6] M. Athans and P. Falb. *Optimal control: an introduction to the theory and its applications*. McGraw-Hill New York, 1966.

[7] A. Aydin. Decentralized structural control approach for Petri nets. *Control and Cybernetics*, 36(1):143–159, 2007.

[8] G. Balbo and M. Silva. *Performance Models for discrete Event Systems with Synchronizations Formalisms and Analysis Techniques*. KRONOS, Zaragoza, 1998.

[9] F. Balduzzi, G. Menga, and A. Giua. First-order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.

[10] F. Basile, A. Giua, and C. Seatzu. Supervisory Control of Petri Nets with Decentralized Monitor Places. In *2007 American Control Conference*, pages 4957 – 4962, New York, USA, 2007.

[11] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *Automatic Control, IEEE Transactions on*, 45(10):1864–1876, 2000.

[12] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[13] G. Berthelot and Lri-Iie. Checking properties of nets using transformations. *Advances in Petri Nets 1985: Lecture Notes in Computer Science*, 222:19–40, 1986.

[14] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[15] J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[16] M. Bramson. Stability of queueing networks. *Probability Surveys*, 5(169-345):1, 2008.

[17] E. Camacho and C. Bordons. *Model Predictive Control*. New York: Springer-Verlag, 2004.

[18] E. Camponogara and L. B. de Oliveira. Distributed optimization for model predictive control of linear-dynamic networks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(6):1331–1338, 2009.

[19] J. Campos, G. Chiola, and M. Silva. Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector. *IEEE Transactions on Software Engineering*, 17:117–125, 2 1991.

[20] J. Campos, J. Colom, H. Jungnitz, and M. Silva. Approximate Throughput Computation of Stochastic Marked Graphs. *IEEE Transactions on Software Engineering*, 20(7):526–535, 1994.

[21] C. Chen. *Linear system theory and design*. Oxford University Press, New York, 1984.

[22] X. Cheng and B. Krogh. Stability-constrained model predictive control. *Automatic Control, IEEE Transactions on*, 46(11):1816–1820, 2001.

[23] G. Chiola, C. Anglano, J. Campos, J. M. Colom, and M. Silva. Operational Analysis of Timed Petri Nets and Application to the Computation of Performance Bounds. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 161–174. Springer, 1995.

[24] P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

[25] R. David and H. Alla. *Discrete, Continuous and Hybrid Petri Nets*. Springer, Berlin, 2004. (Revised 2^{nd} edition, 2010).

[26] M. Dotoli, M. Fanti, A. Giua, and C. Seatzu. First-order hybrid Petri nets. An application to distributed manufacturing systems. *Nonlinear Analysis: Hybrid Systems*, 2(2):408–430, 2008.

[27] M. Dub, G. Pipan, and Z. Hanzálek. Stock optimization of a kanban-based assembly line. In *Proc. of the 12th Int. Conf. on Flexible Automation and Intelligent Manufacturing*, 2002.

[28] J. Ezpeleta, J. M. Couvreur, and M. Silva. A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Coloured Petri Nets. pages 126–147.

[29] E. Fraca, J. Júlvez, and M. Silva. Marking homothetic monotonicity and fluidization of untimed Petri nets. In *11th Int. Workshop on Discrete Event Systems, WODES 2012*, Guadalajara, Mexico, October 2012.

[30] L. Ghomri and H. Alla. Continuous flow systems and control methodology using Hybrid Petri nets. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 419–424. IEEE, 2011.

[31] A. Giua, F. DiCesare, and M. Silva. Petri Net supervisors for generalized mutual exclusion constraints. In *12th IFAC World Congress*, pages 267–270, Sidney, Australia, 1993.

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

[33] L. Habets and J. van Schuppen. A controllability result for piecewise-linear hybrid systems. In *Proceedings of European Control Conference (ECC2001)*, pages 3870–3873, 2001.

[34] L. Habets and J. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40(1):21–35, 2004.

[35] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. *Formal Methods for Computational Systems Biology*, pages 215–264, 2008.

[36] S. Hennequin, D. Lefebvre, and A. El-Moudni. Fuzzy control of variable speed continuous Petri nets. In *38th IEEE Conference on Decision and Control*, volume 2, pages 1352–1356, Phoenix, USA, Dec. 1999.

[37] K. Hiraishi. Performance evaluation of workflows using continuous Petri nets with interval firing speeds. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, 91(11):3219, 2008.

[38] L. E. Holloway, B. H. Krogh, and A. Giua. A Survey of Petri Net Methods for Controlled Discrete Event Systems. *Discrete Event Dynamic Systems*, 7(2):151–190, 1997.

[39] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1):184–201, 1998.

[40] M. V. Iordache and P. J. Antsaklis. Decentralized Supervision of Petri Nets. *IEEE Transactions on Automatic Control*, 51(2):376–391, 2006.

[41] M. V. Iordache and P. J. Antsaklis. *Supervisory control of concurrent systems: A Petri net structural approach*. Birkhauser, 2006.

[42] L. Jiao and T. Cheung. Characterizing liveness monotonicity for weighted Petri nets in terms of siphon-based properties. *International Journal of Foundations of Computer Science*, 14(4):641–658, 2003.

[43] E. Jiménez, J. Júlvez, L. Recalde, and M. Silva. On controllability of timed continuous Petri net systems: the join free case. In *Proc. of the 44th IEEE Conf. on Decision and Control (Joint CDC-ECC)*, Seville, 2005.

[44] X. Jing, 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.

[45] X. Jing, L. Recalde, and M. Silva. Tracking control of timed continuous Petri net systems under infinite servers semantics. In *IFAC World Congress*, 2008.

[46] J. Júlvez. *Algebraic Techniques for the Analysis and Control of Continuous Petri Nets*. PhD thesis, University of Zaragoza, 2004.

[47] J. Júlvez, E. Jiménez, L. Recalde, and M. Silva. On observability and design of observers in timed continuous Petri net systems. *Automation Science and Engineering, IEEE Trans. on*, 5(3):532–537, July 2008.

[48] J. Júlvez, C. Mahulea, and C. Vázquez. SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems*, 6(2):806–817, 2012.

[49] J. Júlvez, L. Recalde, and M. Silva. On reachability in autonomous continuous petri net systems. In *Applications and Theory of Petri Nets 2003*, pages 221–240. Springer, 2003.

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

[51] R. Kara, M. Ahmane, J. Loiseau, and S. Djennoune. Constrained regulation of continuous Petri nets. *Nonlinear Analysis: Hybrid Systems*, 3(4):738–748, 2009.

[52] S. Keerthi and E. Gilbert. Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints. *IEEE Transactions on Automatic Control*, 32(5):432–435, 1987.

[53] C. Kelling. Timenet-sim-a parallel simulator for stochastic petri nets. In *Simulation Symposium, 1995., Proceedings of the 28th Annual*, pages 250–258. IEEE, 1995.

[54] A. Lazar. Optimal flow control of a class of queueing networks in equilibrium. *IEEE Transactions on Automatic Control*, 28(11):1001 – 1007, 1983.

[55] D. Lefebvre. Optimal flow control for manufacturing systems modelled by continuous Petri nets. In *39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[56] D. Lefebvre. About the stochastic and continuous Petri nets equivalence in the long run. *Nonlinear Analysis: Hybrid Systems*, 5(3):394–406, 2011.

[57] D. Lefebvre. Approximation of the asymptotic mean marking of SPNs with contPNs. *Nonlinear Analysis: Hybrid Systems*, 6(4):972–987, 2012.

[58] D. Lefebvre, C. Delherm, E. Leclercq, and F. Druaux. Some contributions with Petri nets for the modelling, analysis and control of HDS. *Nonlinear Analysis: Hybrid Systems*, 1:451–465, 2007.

[59] D. Lefebvre and E. Leclercq. Piecewise constant timed continuous PNs for the steady state estimation of stochastic PNs. *Discrete Event Dynamic Systems*, 22(2):179–196, 2012.

[60] Z. Li, M. Zhou, and N. Wu. A Survey and Comparison of Petri Net-Based Deadlock Prevention Policies for Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man and Cybernetics - PART C: APPLICATIONS AND REVIEWS*, 38(2):173–188, 2008.

[61] Z. W. Li and M. C. Zhou. Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(1):38–51, 2004.

[62] G. F. List and M. Cetin. Modeling traffic signal control using Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):177–187, 2004.

[63] C. Mahulea. *Timed Continuous Petri Nets: Quantitative Analysis, Observability and Control*. PhD thesis, University of Zaragoza, 2007.

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

[65] C. Mahulea, A. Ramírez, 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.

[66] 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.

[67] C. Mahulea, L. Recalde, and M. Silva. Observability of continuous Petri nets with infinite server semantics. *Nonlinear Analysis: Hybrid Systems*, 4(2):219–232, 2010.

[68] C. Mahulea, C. Seatzu, M. Cabasino, and M. Silva. Fault Diagnosis of Discrete-Event Systems using Continuous Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 42(4):970 – 984, 2012.

[69] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with generalized stochastic Petri nets*. 1995, John Wiley and Sons.

[70] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[71] J. Merseguer and J. Campos. Software performance modeling using UML and Petri nets. *Performance Tools and Applications to Networked Systems*, pages 265–289, 2004.

[72] D. Mitra. Stochastic theory of a fluid model of producers and consumers coupled by a buffer. *Advances in Applied Probability*, pages 646–676, 1988.

[73] M. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9):913–917, 1982.

[74] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[75] C. Pérez-Jiménez, J. Campos, and M. Silva. Approximate Throughput Computation of Stochastic Weighted T-Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(3):431 – 444, 2007.

[76] C. A. Petri. *Communication With Automata*. Supplement 1 to Technical Report RADC-TR-65-377, Vol. 1, Griffiss Air Force Base, New York 1966.

[Originally published in German: Kommunikation mit Automaten, University of Bonn], 1962.

[77] L. S. Pontryagin. *Mathematical theory of optimal processes*, volume 4. CRC, 1987.

[78] S. Rajagopal, V. G. Kulkami, and J. Shaler Stidham. Optimal flow control of a stochastic fluid-flow system. *IEEE Journal on Selected Areas in Communications*, 13(7):1219 – 1228, 1995.

[79] L. Recalde, S. Haddad, and M. Silva. Continuous Petri Nets: Expressive Power and Decidability Issues. *Int. Journal of Foundations of Computer Science*, 21(2):235–256, 2010.

[80] L. Recalde and M. Silva. Petri nets fluidification revisited semantics and steady state. *APII JESA*, 35(4):435–449, 2001.

[81] L. Recalde, E. Teruel, and M. Silva. Modeling and Analysis of Sequential Processes that Cooperate Through Buffers. *IEEE Transactions on Robotics and Automation*, 14(2):267–276, 1998.

[82] L. Recalde, E. Teruel, and M. Silva. On Linear Algebraic Techniques for Liveness Analysis of P/T Systems. *Journal of Circuits Systems and Computers*, 8(1):223–265, 1998.

[83] L. Recalde, E. Teruel, and M. Silva. Autonomous Continuous P/T systems. In J. K. S. Donatelli, editor, *Application and Theory of Petri Nets 1999*, volume 1639 of *Lecture Notes in Computer Science*, pages 107–126. Springer, 1999.

[84] R. Roberto, R. Antonio, M. J. Alejandro, and R. Javier. Control Of Metabolic Systems Modeled with Timed Continuous Petri Nets. In *Proceedings of the Workshops of the 31st International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2010) and of the 10th International Conference on Application of Concurrency to System Design (ACSD 2010)*, pages 87–102, Braga, Portugal, June 2010.

[85] R. Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, 19(5):723–731, 2009.

[86] K. Shin and N. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[87] M. Silva, J. Júlvez, C. Mahulea, and C. R. Vázquez. On fluidization of discrete event models: observation and control of continuous Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 21(4):427–497, 2011.

[88] M. Silva and L. Recalde. Petri nets and integrality relaxations: A view of continuous Petri net models. *IEEE Trans. on Systems, Man, and Cybernetics*, 32(4):314–327, 2002.

[89] 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.

[90] M. Silva, E. Teruel, and J. M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Net Systems. *Lecture Notes in Computer Science*, 1491:309–373, 1998.

[91] B. Stewart, A. Venkat, J. Rawlings, S. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59(8):460–469, 2010.

[92] E. Teruel, P. Chrzastowski-Wachtel, J. Colom, and M. Silva. On weighted T-systems. In K. Jensen, editor, *Application and Theory of Petri nets. LNCS*, volume 616, pages 348–367, Berlin, Germany, 1992.

[93] 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.

[94] E. Teruel and M. Silva. Structure theory of equal conflict systems. *Theoretical Computer Science*, 153(1):271–300, 1996.

[95] K. Trivedi and V. Kulkarni. FSPNs: fluid stochastic Petri nets. *Application and Theory of Petri Nets 1993*, pages 24–31, 1993.

[96] C. Vázquez. *Fluidization, Controllability and Control of Timed Continuous Petri Nets*. PhD thesis, University of Zaragoza, 2011.

[97] C. Vázquez, A. Ramírez, L. Recalde, and M. Silva. On Controllability of Timed Continuous Petri Nets. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computational and Control (HSCC'08)*, volume 4981, pages 528—-541, Saint Louis, USA, July 2008. Springer Berlin / Heidelberg.

[98] C. Vázquez and M. Silva. Performance control of Markovian Petri nets via fluid models: A stock-level control example. In *IEEE International Conference on Automation Science and Engineering*, pages 30–36, 2009.

[99] C. Vázquez and M. Silva. Piecewise-linear constrained control for timed continuous Petri nets. In *Proc. of the 48th IEEE Conf. on Decision and Control (CDC)*, 2009.

[100] C. Vázquez and M. Silva. Timing and liveness in continuous Petri nets. *Automatica*, 47:283–290, 2011.

[101] C. Vázquez and M. Silva. Stochastic Continuous Petri Nets: An approximation of Markovian Net Models. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(3):641–653, 2012.

[102] C. Vázquez, J. van Schuppen, and M. Silva. A modular-coordinated control for continuous Petri nets. In *18th IFAC World Congress*, Milan, Italy, August 2011.

[103] L. Wang, C. Mahulea, J. Júlvez, , and M. Silva. Minimum-time Decentralized Control of Choice-Free Continuous Petri Nets. *Nonlinear Analysis: Hybrid Systems*, 7(1):39–53, 2013.

[104] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Minimum-time Control for Structurally Persistent Continuous Petri Nets. In *49th IEEE Conference on Decision and Control*, pages 2771–2776, Atlanta, Georgia USA, December 2010.

[105] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Decentralized Control of Large Scale Systems Modeled with Continuous Marked Graphs. In *18th IFAC World Congress*, Milan, Italy, Aug. 2011.

[106] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Control of continuous Petri nets using ON/OFF based method. In *WODES12: 11th Int. Workshop on Discrete Event Systems*, pages 47–52, Guadalajara, Mexico, October 2012.

[107] L. Wang, C. Mahulea, and M. Silva. Distributed Model Predictive Control of Timed Continuous Petri nets. In *52nd IEEE Conference on Decision and Control (CDC2013)*, Italy, December 2013. submitted.

[108] L. Wang, C. Mahulea, and M. Silva. Minimum-Time Flow Control of Timed Continuous Choice-Free Nets. In *ECC13: European Control Conference*, Zurich, Switzerland, July 2013. to appear.

[109] W. Wang, M. Palaniswami, and S. Low. Optimal flow control and routing in multi-path networks. *Performance Evaluation*, 52(2):119–132, 2003.

[110] G. Xie and L. Wang. Controllability and stabilizability of switched linear-systems. *Systems & Control Letters*, 48(2):135–155, 2003.

[111] A. Zimmermann, D. Rodríguez, and M. Silva. A two phase optimization method for Petri net models of manufacturing systems. *Journal of Intelligent Manufacturing*, 12(5):409–420, 2001.

Appendix A

Simulation Results

Table A.1: Simulation results of the net system in Fig. 4.11. Setting **s**.1): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	101	847	
ON/OFF+	94	41	
B-ON/OFF	91	130	$d = 1$
B-ON/OFF	91	136	$d = 2$
B-ON/OFF	94	141	$d = 5$
B-ON/OFF	94	139	$d = 10$
B-ON/OFF	94	138	$d = 15$
B-ON/OFF	94	142	$d = 20$
MPC-ON/OFF	91	1,159	$N = 1$, r=1000, q=1000
MPC-ON/OFF	91	877	$N = 1$, r=1000, q=100
MPC-ON/OFF	91	955	$N = 1$, r=1000, q=1
MPC-ON/OFF	95	5,546	$N = 3$, r=1000, q=1000
MPC-ON/OFF	95	5,619	$N = 3$, r=1000, q=100
MPC-ON/OFF	97	7,678	$N = 3$, r=1000, q=1
MPC-ON/OFF	94	11,188	$N = 5$, r=1000, q=1000
MPC-ON/OFF	95	11,369	$N = 5$, r=1000, q=100
MPC-ON/OFF	91	11,811	$N = 5$, r=1000, q=1
MPC-ON/OFF	93	54,311	$N = 10$, r=1000, q=1000
MPC-ON/OFF	94	53,818	$N = 10$, r=1000, q=10
MPC-ON/OFF	94	50,784	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF ($d = 1$ or 2) or the MPC-ON/OFF controller (with $N =$ 1 or 5). For the B-ON/OFF controller, smaller numbers of time steps are obtained using smaller values of d (a small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.2: Simulation results of the net system in Fig. 4.11. Setting **s**.2): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.001\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.301\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	176	465	
ON/OFF+	954	410	
B-ON/OFF	132	197	$d = 1$
B-ON/OFF	132	192	$d = 2$
B-ON/OFF	196	287	$d = 5$
B-ON/OFF	258	393	$d = 10$
B-ON/OFF	290	437	$d = 15$
B-ON/OFF	335	504	$d = 20$
MPC-ON/OFF	165	1,942	$N = 1$, r=1000, q=1000
MPC-ON/OFF	158	1,687	$N = 1$, r=1000, q=100
MPC-ON/OFF	149	2,697	$N = 1$, r=1000, q=1
MPC-ON/OFF	165	8,005	$N = 3$, r=1000, q=1000
MPC-ON/OFF	164	8,048	$N = 3$, r=1000, q=100
MPC-ON/OFF	163	7,448	$N = 3$, r=1000, q=1
MPC-ON/OFF	162	14,638	$N = 5$, r=1000, q=1000
MPC-ON/OFF	162	14,593	$N = 5$, r=1000, q=100
MPC-ON/OFF	145	16,240	$N = 5$, r=1000, q=1
MPC-ON/OFF	159	80,193	$N = 10$, r=1000, q=1000
MPC-ON/OFF	159	80,417	$N = 10$, r=1000, q=10
MPC-ON/OFF	159	86,426	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF controller ($d = 1$ or 2). For the B-ON/OFF controller, smaller numbers of time steps are obtained using smaller values of d (a small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly. For the MPC-ON/OFF controller, smaller numbers of time steps are obtained by using larger weights on matrix $\boldsymbol{R}$ and smaller weights on matrix $\boldsymbol{Q}$; the numbers of time steps are slightly reduced by using larger N.

Table A.3: Simulation results of the net system in Fig. 4.11. Setting **s**.3): $\Theta = 0.1$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 3\ 2.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	94	352	
ON/OFF+	76	34	
B-ON/OFF	78	121	$d = 1$
B-ON/OFF	78	120	$d = 2$
B-ON/OFF	78	121	$d = 5$
B-ON/OFF	76	114	$d = 10$
B-ON/OFF	76	114	$d = 15$
B-ON/OFF	76	115	$d = 20$
MPC-ON/OFF	94	958	$N = 1$, r=1000, q=1000
MPC-ON/OFF	94	1,010	$N = 1$, r=1000, q=100
MPC-ON/OFF	94	1,067	$N = 1$, r=1000, q=1
MPC-ON/OFF	93	9,800	$N = 3$, r=1000, q=1000
MPC-ON/OFF	92	5,178	$N = 3$, r=1000, q=100
MPC-ON/OFF	93	7,394	$N = 3$, r=1000, q=1
MPC-ON/OFF	90	13,958	$N = 5$, r=1000, q=1000
MPC-ON/OFF	84	14,726	$N = 5$, r=1000, q=100
MPC-ON/OFF	84	15,482	$N = 5$, r=1000, q=1
MPC-ON/OFF	75	52,803	$N = 10$, r=1000, q=1000
MPC-ON/OFF	78	57,987	$N = 10$, r=1000, q=10
MPC-ON/OFF	76	56,352	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the MPC-ON/OFF controller (with $N = 10$). The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.4: Simulation results of the net system in Fig. 4.11. Setting **s**.4): $\Theta = 0.02$, $\boldsymbol{m}_0 = [1\ 2\ 1.4\ 1.5\ 1.1\ 1.1\ 1.1\ 5\ 1.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 1.7\ 1.2\ 1.2\ 1.5\ 1.3\ 3\ 3.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	122	2,546	
ON/OFF+	126	55	
B-ON/OFF	128	200	$d = 1$
B-ON/OFF	128	195	$d = 2$
B-ON/OFF	128	195	$d = 5$
B-ON/OFF	126	191	$d = 10$
B-ON/OFF	126	192	$d = 15$
B-ON/OFF	126	195	$d = 20$
MPC-ON/OFF	133	1,458	$N = 1$, r=1000, q=1000
MPC-ON/OFF	133	1,441	$N = 1$, r=1000, q=100
MPC-ON/OFF	115	1,283	$N = 1$, r=1000, q=1
MPC-ON/OFF	131	5,855	$N = 3$, r=1000, q=1000
MPC-ON/OFF	131	6,943	$N = 3$, r=1000, q=100
MPC-ON/OFF	131	6,386	$N = 3$, r=1000, q=1
MPC-ON/OFF	128	13,580	$N = 5$, r=1000, q=1000
MPC-ON/OFF	130	13,539	$N = 5$, r=1000, q=100
MPC-ON/OFF	130	16,016	$N = 5$, r=1000, q=1
MPC-ON/OFF	126	92,728	$N = 10$, r=1000, q=1000
MPC-ON/OFF	126	90,033	$N = 10$, r=1000, q=10
MPC-ON/OFF	125	90,781	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the approaching minimum-time controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.5: Simulation results of the net system in Fig. 8.5. $\boldsymbol{m}_0 = [0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 4\ 4\ 2\ 5\ 1\ 2\ 1\ 2]^T$, $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.8125\ 1.083\ 0.8125\ 0.55]^T$, $\boldsymbol{\lambda} = [10\ 10\ 2\ 2.5\ 2.5\ 10\ 10\ 1.25\ 2.5\ 2.5\ 2.5\ 0.5]^T$, $\boldsymbol{\sigma} = [2.345\ 2.368\ 2.08\ 1.87\ 1.66\ 2.578\ 2.6\ 2.08\ 1.87\ 1.66\ 1.45\ 0]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	311	88,505	
ON/OFF+	457	275	
B-ON/OFF	217	443	$d = 1$
B-ON/OFF	209	426	$d = 2$
B-ON/OFF	210	431	$d = 5$
B-ON/OFF	216	448	$d = 10$
B-ON/OFF	234	491	$d = 15$
B-ON/OFF	280	596	$d = 20$
MPC-ON/OFF	220	4,815	$N = 1$, r=1000, q=1000
MPC-ON/OFF	221	4,994	$N = 1$, r=1000, q=100
MPC-ON/OFF	221	5,023	$N = 1$, r=1000, q=1
MPC-ON/OFF	220	29,115	$N = 3$, r=1000, q=1000
MPC-ON/OFF	220	28,947	$N = 3$, r=1000, q=100
MPC-ON/OFF	220	30,017	$N = 3$, r=1000, q=1
MPC-ON/OFF	219	92,156	$N = 5$, r=1000, q=1000
MPC-ON/OFF	219	94,743	$N = 5$, r=1000, q=100
MPC-ON/OFF	219	102,622	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF controller (with $d = 2$). For the B-ON/OFF controller, smaller numbers of time steps are obtained with smaller values of d (a small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly), but an exception happened when $d = 1$. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to time horizon N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.6: Simulation results of the net system in Fig. 8.6. $\boldsymbol{m}_0 = [0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 4\ 4\ 2\ 5\ 1\ 2\ 1\ 2]^T$, $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.58\ 1.083\ 1.045\ 0.55]^T$, $\boldsymbol{\lambda} = [10\ 10\ 2\ 2.5\ 2.5\ 10\ 10\ 1.25\ 2.5\ 2.5\ 2.5\ 0.5]^T$, $\boldsymbol{\sigma} = [2.345\ 2.368\ 2.08\ 1.87\ 1.66\ 2.578\ 2.6\ 2.08\ 1.87\ 1.66\ 1.45\ 0]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	316	55,608	
ON/OFF+	223	108	
B-ON/OFF	224	427	$d = 1$
B-ON/OFF	223	420	$d = 2$
B-ON/OFF	223	421	$d = 5$
B-ON/OFF	223	421	$d = 10$
B-ON/OFF	223	421	$d = 15$
B-ON/OFF	223	419	$d = 20$
MPC-ON/OFF	229	4,967	$N = 1$, r=1000, q=1000
MPC-ON/OFF	229	4,846	$N = 1$, r=1000, q=100
MPC-ON/OFF	228	5,018	$N = 1$, r=1000, q=1
MPC-ON/OFF	230	31,281	$N = 3$, r=1000, q=1000
MPC-ON/OFF	229	30,536	$N = 3$, r=1000, q=100
MPC-ON/OFF	229	31,914	$N = 3$, r=1000, q=1
MPC-ON/OFF	228	96,231	$N = 5$, r=1000, q=1000
MPC-ON/OFF	228	97,993	$N = 5$, r=1000, q=100
MPC-ON/OFF	228	99,749	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the ON/OFF+ controller or the B-ON/OFF controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to time horizon N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.7: Simulation results of the net system in Fig. 8.8. $\boldsymbol{m}_0 = [10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 5\ 0.1\ 0.1\ 5\ 5\ 5\ 5\ 5\ 5\ 5\ 8\ 0.1\ 2\ 2\ 2\ 2\ 2]^T$, $\boldsymbol{m}_f = [5.78\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 1.024\ 0.05122\ 4.756\ 2.049\ 0.05122\ 0.5122\ 0.05122\ 5.78\ 1.024\ 0.05122\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 4.124\ 2.049\ 0.2561\ 4.637\ 4.124\ 3.1\ 4.637\ 4.124\ 4.637\ 4.124\ 4.844\ 7.844\ 0.2561\ 0.7634\ 0.05122\ 0.1512\ 0.1512\ 0.05122]^T$, $\boldsymbol{\lambda} = [50\ 5\ 50\ 2.5\ 50\ 50\ 1.25\ 50\ 5\ 50\ 2.5\ 50\ 5;50\ 2.5\ 2.5\ 50\ 1.25\ 10\ 1.25\ ]^T$, $\boldsymbol{\sigma} = [4.22\ 3.807\ 3.856\ 2.932\ 2.98\ 5.244\ 3.295\ 3.344\ 2.932\ 4.22\ 3.295\ 3.344\ 2.932\ 2.98\ 2.056\ 2.056\ 2.105\ 0.1561\ 0\ 0.1561]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	300	138,826	
ON/OFF+	249	213	
B-ON/OFF	131	470	$d = 1$
B-ON/OFF	131	471	$d = 2$
B-ON/OFF	131	472	$d = 5$
B-ON/OFF	131	475	$d = 10$
B-ON/OFF	150	550	$d = 15$
B-ON/OFF	249	892	$d = 20$
MPC-ON/OFF	146	10,117	$N = 1$, r=1000, q=1000
MPC-ON/OFF	150	10,377	$N = 1$, r=1000, q=100
MPC-ON/OFF	149	10,852	$N = 1$, r=1000, q=1
MPC-ON/OFF	134	86,935	$N = 3$, r=1000, q=1000
MPC-ON/OFF	133	84,778	$N = 3$, r=1000, q=100
MPC-ON/OFF	134	83,968	$N = 3$, r=1000, q=1
MPC-ON/OFF	132	284,205	$N = 5$, r=1000, q=1000
MPC-ON/OFF	131	284,859	$N = 5$, r=1000, q=100
MPC-ON/OFF	132	276,821	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF ($d \leq 10$) or the MPC-ON/OFF controller (with $N = 5$). For the B-ON/OFF controller, smaller numbers of time steps are obtained with smaller values of d, when $d \leq 10$ the best result is obtained. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.8: Simulation results of the net system in Fig. 8.10. $\boldsymbol{m}_0 = [5\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 7\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 3\ 3\ 3\ 1\ 1\ 1\ 1]^T$, $\boldsymbol{m}_f = [3.4\ 1.3\ 0.4\ 0.2\ 7.6\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 3.1\ 0.8\ 0.4\ 0.4\ 0.4\ 2.4\ 2\ 0.8\ 0.4\ 0.7\ 0.4\ 0.4\ 0.4]^T$, $\lambda = [0.3333\ 0.5\ 0.5\ 0.5\ 0.5\ 1\ 0.5\ 0.5\ 0.5\ 0.5\ 0.25\ 0.5\ 0.5\ 1\ 0.5\ 0.5\ 0.5\ 0.5\ 0.5\ 0.25]^T$, $\boldsymbol{\sigma} = [2.4\ 1.2\ 0.9\ 0.6\ 0.3\ 0\ 0.9\ 0.6\ 0.3\ 0\ 1.6\ 0.4\ 0.1\ 0\ 3.9\ 1.6\ 1.3\ 1\ 0.7\ 0]^T$, $\Theta = 0.03$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	279	60,691	
ON/OFF+	301	271	
B-ON/OFF	301	1,151	$d = 1$
B-ON/OFF	301	1,144	$d = 2$
B-ON/OFF	301	1,151	$d = 5$
B-ON/OFF	301	1,147	$d = 10$
B-ON/OFF	301	1,149	$d = 15$
B-ON/OFF	301	1,157	$d = 20$
MPC-ON/OFF	320	17,873	$N = 1$, r=1000, q=1000
MPC-ON/OFF	320	19,783	$N = 1$, r=1000, q=100
MPC-ON/OFF	320	21,440	$N = 1$, r=1000, q=1
MPC-ON/OFF	317	126,208	$N = 3$, r=1000, q=1000
MPC-ON/OFF	315	129,551	$N = 3$, r=1000, q=100
MPC-ON/OFF	316	137,814	$N = 3$, r=1000, q=1
MPC-ON/OFF	310	431,671	$N = 5$, r=1000, q=1000
MPC-ON/OFF	310	425,928	$N = 5$, r=1000, q=100
MPC-ON/OFF	310	443,439	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the approaching minimum-time controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.