



**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 artificiales, 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 reducción 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 prácticos.

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 *persistence* 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 \mathbf{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

CONTENTS

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
B	More discussions of the MPC-ON/OFF controller	171

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 parallel 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 \mathbf{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:

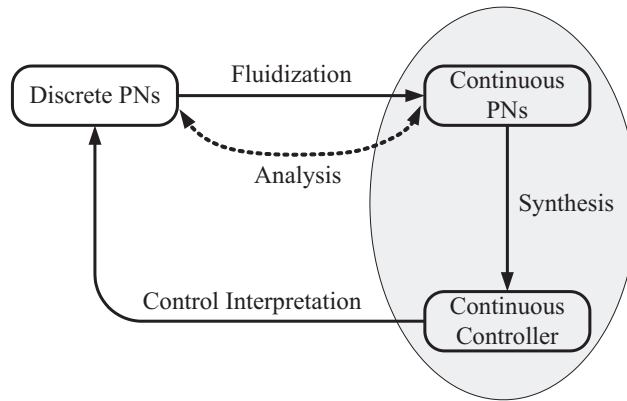


Figure 1.1: The sketch of the research field

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 *persistence* 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 $\langle \mathcal{N}, \mathbf{M}_0 \rangle$, in which $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ is a net structure, where:

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

Let $p_i, i = 1, \dots, |P|$ and $t_j, j = 1, \dots, |T|$ denote the places and transitions. $\mathbf{Pre}[p_i, t_j] = w_1$ and $\mathbf{Post}[p_i, t_j] = w_2$ 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_i 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 \mathbf{M} and the marking of place p_i is represented as $\mathbf{M}[p_i]$. In (discrete) PNs one transition t_j is enabled at marking \mathbf{M} if each of its input place $p_i \in \bullet t_j$ fulfills $\mathbf{M}[p_i] \geq \mathbf{Pre}[p_i, t_j]$. The enabling degree of transition t_j at marking \mathbf{M} is defined as:

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

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

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

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 $\boldsymbol{\sigma}$.

2.2. Autonomous (untimed) continuous Petri nets

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.

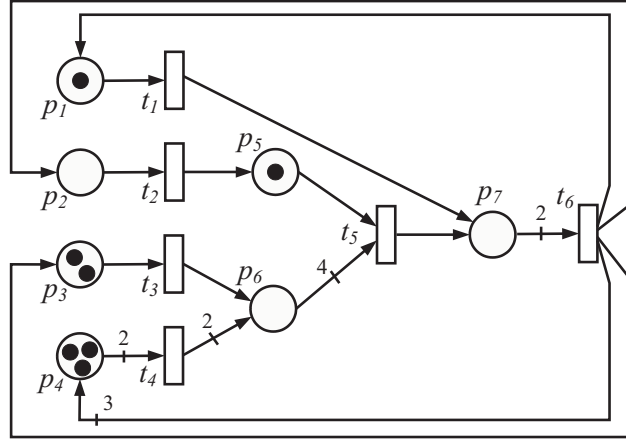


Figure 2.1: A simple PN that models an assembly system, initial state $\mathbf{M}_0 = [1 \ 0 \ 2 \ 3 \ 1 \ 0 \ 0]^T$

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

Initial state	Size of the reachability set
\mathbf{M}_0	54
$2 \cdot \mathbf{M}_0$	1,685
$3 \cdot \mathbf{M}_0$	10,354
$4 \cdot \mathbf{M}_0$	37,722
$5 \cdot \mathbf{M}_0$	103,914
\dots	\dots
$10 \cdot \mathbf{M}_0$	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 $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ where $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ 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, $\mathbf{m}_0 \in \mathbb{R}_{\geq 0}^{|P|}$. In CPN systems, the markings are denoted by \mathbf{m} , distinguishing with \mathbf{M} for the markings in discrete models.

In CPN systems, a transition t is *enabled* at \mathbf{m} if for every $p \in {}^\bullet t$, $\mathbf{m}[p] > 0$, 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 \mathbf{m} , which is defined as:

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

An enabled transition t_j can fire in any real amount α , with $0 < \alpha \leq enab(t_j, \mathbf{m})$, leading to a new state $\mathbf{m}' = \mathbf{m} + \alpha \cdot \mathbf{C}[P, t_j]$. Similar to discrete systems, a reachable marking from \mathbf{m}_0 through a finite sequence σ is included in the state (fundamental) equation:

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

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 ${}^\bullet t \cap {}^\bullet t' \neq \emptyset \Rightarrow \mathbf{Pre}[P, t] = \mathbf{Pre}[P, t']$.
- Mono-T-semiflow (MTS) [19]: conservative and has a unique minimal T-semiflow whose support contains all the transitions.

2.2. Autonomous (untimed) continuous Petri nets

2.2.3 Basic structural concepts

The *support* of a vector, $\mathbf{v} \geq \mathbf{0}$, is $\|\mathbf{v}\| = \{v_i | v_i > 0\}$, the set of positive elements of \mathbf{v} . Right ($\mathbf{C} \cdot \mathbf{x} = 0$) and left ($\mathbf{y} \cdot \mathbf{C} = 0$) 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 $\exists \mathbf{y} > \mathbf{0}$, s.t. $\mathbf{y}^T \cdot \mathbf{C} = \mathbf{0}$, the net is said to be *conservative*, and when $\exists \mathbf{x} > \mathbf{0}$ s.t. $\mathbf{C} \cdot \mathbf{x} = \mathbf{0}$, the net is said to be *consistent*.

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

- *conservation laws*: a set of equations $\mathbf{y}^T \cdot \mathbf{m}_0 = \mathbf{y}^T \cdot \mathbf{m}$, which hold for an arbitrary initial marking \mathbf{m}_0 and every reachable marking $\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$, $\mathbf{m}, \boldsymbol{\sigma} \geq \mathbf{0}$.
- *conservative component*: the P-subnet generated by the support of \mathbf{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 $\exists \mathbf{x} \succeq \mathbf{0}$ s.t. $\mathbf{C} \cdot \mathbf{x} = \mathbf{0}$, and \mathbf{x} is fireable from \mathbf{m} then, by the state equation, $\mathbf{m} \xrightarrow{\boldsymbol{\sigma}} \mathbf{m}$ with $\boldsymbol{\sigma}$ being a firing sequence and the corresponding firing count vector is equal to \mathbf{x} .

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

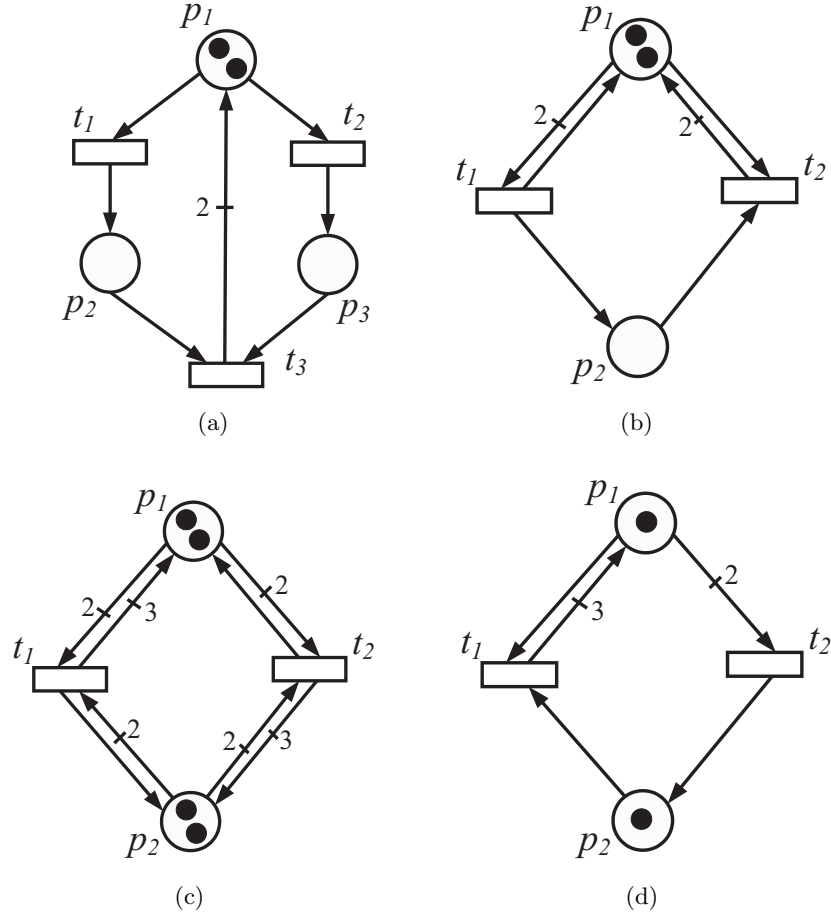


Figure 2.2: Some simple PN systems

2.2.4 Reachability and lim-reachability

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

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

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

2.2. Autonomous (untimed) continuous Petri nets

Example 2.2.5. *Let us consider the system in Fig. 2.2(b). At the initial marking $\mathbf{m}_0 = [2 \ 0]^T$, 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 $\mathbf{m}_1 = [1 \ 1]^T$, from which both transitions (t_1 and t_2) are enabled. From marking \mathbf{m}_1 , if t_1 fires an amount equal to $\text{enab}(t_1, \mathbf{m}_1) = \frac{1}{2}$, the system reaches $\mathbf{m}_2 = [\frac{1}{2} \ \frac{3}{2}]^T$. 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 $\mathbf{m} = [0 \ 2]^T$, 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 $\text{lim-RS}(\mathcal{N}, \mathbf{m}_0)$:

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

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

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

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

Assuming an initial marking of non-negative integers of a continuous system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$, if \mathbf{m} is a marking that is reached by firing transitions in discrete amounts, i.e., as if the system was discrete, then \mathbf{m} is also reachable by the system as continuous just by applying the same firing sequence. Thus $\text{RS}_D(\mathcal{N}, \mathbf{M}_0) \subseteq \text{RS}(\mathcal{N}, \mathbf{m}_0)$ where $\mathbf{M}_0 = \mathbf{m}_0$ and $\text{RS}_D(\mathcal{N}, \mathbf{M}_0)$ 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 $\text{lim-RS}(\mathcal{N}, \mathbf{m}_0)$ by using some linear inequality systems, which can be easily checked, in polynomial time:

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

- \mathbf{m} is lim-reachable.
- $\exists \boldsymbol{\sigma} \geq 0$, such that $\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$.
- $\mathbf{B}_y \cdot \mathbf{m} = \mathbf{B}_y \cdot \mathbf{m}_0, \mathbf{m} \geq 0$, where \mathbf{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 $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ 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 \mathbf{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
- $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is bounded

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

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

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 $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ be a continuous PN system:*

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

If we consider $\text{lim-RS}(\mathcal{N}, \mathbf{m}_0)$, 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

2.2. Autonomous (untimed) continuous Petri nets

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 $\mathbf{m}_d = [0 \ 1.5]^T$ 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 $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ 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 $\langle \mathcal{N}, \mathbf{m}_0 \rangle$, 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 \mathbf{m}_0 from which it is implicit.*

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

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

1. $\exists \mathbf{y} \geq \mathbf{0}$, such that $\mathbf{C}[p, T] \geq \mathbf{y} \cdot \mathbf{C}[P, T]$ and $\mathbf{y}[p] = 0$
2. $\nexists \mathbf{x} \geq \mathbf{0}$, such that $\mathbf{C}[P, T] \cdot \mathbf{x} \geq \mathbf{0}$ and $\mathbf{C}[p, T] \cdot \mathbf{x} < \mathbf{0}$

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

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

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: $\mathbf{m}(\tau) = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}(\tau)$, which, through time differentiation, becomes $\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$. The derivative of the firing sequence $\mathbf{f}(\tau) = \dot{\boldsymbol{\sigma}}(\tau)$ is called the (*firing*) *flow*, and leads to the following equation for the dynamics of TCPN systems:

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

Depending on how the flow \mathbf{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 $\lambda[t_j]$ (or denoted by λ_j) is assigned to each transition t_j . For *finite server semantics*, if the markings of the input places of

2.3. Timed continuous Petri nets

t_j are strictly greater than zero (*strongly enabled*), its flow will be constant, equal to $\lambda[t_j]$, 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, $\lambda[t_j]$ 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:

$$f[t_j] = \begin{cases} \lambda[t_j], & \text{if } \forall p_i \in \bullet t_j, m_i > 0 \\ \min \left\{ \min_{p_i \in \bullet t_j | m_i = 0} \left\{ \sum_{t_q \in \bullet p_i} \frac{f[t_q] \cdot \text{Post}[t_q, p_i]}{\text{Pre}[p_i, t_j]} \right\}, \lambda[t_j] \right\}, & \text{otherwise} \end{cases} \quad (2.7)$$

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:

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

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 \mathbf{m} , is known as *configuration* at \mathbf{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:

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

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

Let us notice that regions are disjoint except on the borders and the reachability set $RS(\mathcal{N}, \mathbf{m}_0)$ of a TCPN system can be partitioned according to the configurations and inside each obtained *convex region* $\mathcal{R}_i(\mathcal{N}, \mathbf{m}_0)$ 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:

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

where $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ is a diagonal $|T| \times |T|$ matrix containing the firing rates of transitions and $\mathbf{\Pi}(\mathbf{m}) = \mathbf{\Pi}_k$ is the configuration matrix associated to \mathcal{R}_k (if \mathbf{m} is on the border of two regions \mathcal{R}_1 and \mathcal{R}_2 , any operation mode with $\mathbf{C}\mathbf{\Lambda}\mathbf{\Pi}_1$ or $\mathbf{C}\mathbf{\Lambda}\mathbf{\Pi}_2$ 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 \mathbf{f}_{ss} be the flow vector of the timed system in the steady state, $\mathbf{f}_{ss} = \lim_{\tau \rightarrow \infty} \mathbf{f}(\tau)$, from (2.6) $\dot{\mathbf{m}} = \mathbf{C} \cdot \mathbf{f}_{ss} = 0$ 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:

2.3. Timed continuous Petri nets

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

- $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ is timed-deadlock-free if $\mathbf{f}_{ss} \neq \mathbf{0}$;
- $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ is timed-live if $\mathbf{f}_{ss} > \mathbf{0}$;
- $\langle \mathcal{N}, \lambda \rangle$ is structurally timed-live if $\exists \mathbf{m}_0$ such that $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ 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 $\lambda = [1 \ 1 \ 2]^T$ (in particular $\mathbf{f}_{ss} = [1 \ 1 \ 1]^T$). 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]

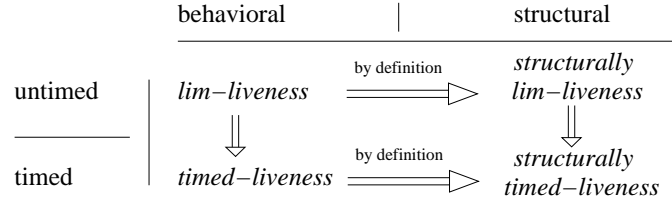


Figure 2.3: Relationships among liveness definitions for continuous models [87]

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 \mathbf{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 :

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

where \mathbf{m}_{ss} is the steady-state marking. Due to the *minimum* operator, problem (2.11) is non linear and a branch & bound algorithm was proposed 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].

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

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 \mathbf{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 (\mathbf{f}_{ss}) in the steady state, the marking in the steady state (\mathbf{m}_{ss}), and the initial marking (\mathbf{m}_0). The profit function can be represented, in linear terms, like: $\mathbf{g} \cdot \mathbf{f}_{ss} - \mathbf{w} \cdot \mathbf{m}_{ss} - \mathbf{b} \cdot \mathbf{m}_0$, where \mathbf{g} is a gain vector w.r.t. the flow; \mathbf{w} is the cost vector due to immobilization to maintain the production flow, e.g. due to the

2.3. Timed continuous Petri nets

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

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

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

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 \mathbf{m}_0 appears as a variable and that the linear cost-constraints associated to \mathbf{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 \mathbf{b} such that a certain cycle time $\Gamma = 1/\mathbf{f}_{ss}[t_j]$ is guaranteed. This optimization problem can be solved by means of the following LPP [89]:

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

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 caused 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 from \mathbf{m}_0 to $10 \cdot \mathbf{m}_0$, 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.*

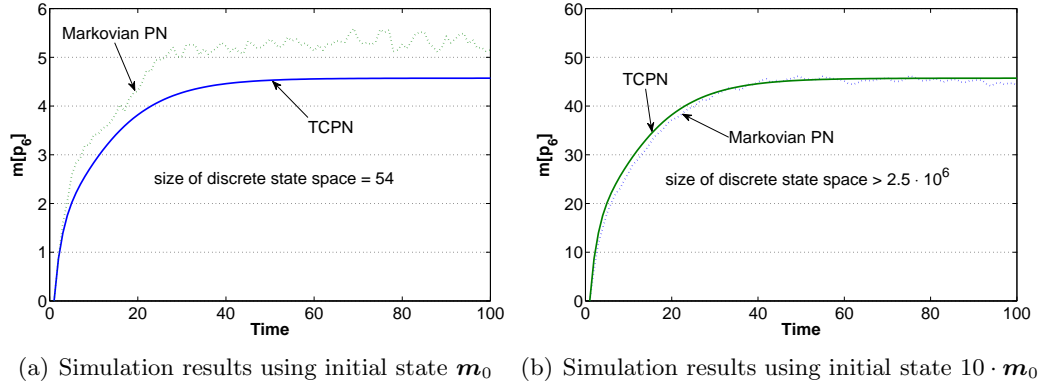


Figure 2.4: Simulations: a discrete PN and its fluid model

There are two main reasons that may introduce errors to the fluid models: the *weights* on arcs and *join* transitions (rendez-vous). Let \mathbf{M} be the marking of the original discrete PN and \mathbf{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 $\mathbf{m} \sim E(\mathbf{M})$, where $E(\mathbf{M})$ refers to the *expectation* of \mathbf{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 $E(enab(t_j, \mathbf{M})) = E(\lfloor \mathbf{M}[p_i]/w \rfloor)$; while in the TCPN, $enab(t_j, \mathbf{m}) = \mathbf{m}[p_i]/w \sim E(\mathbf{M}[p_i])/w$. Clearly, due to the operation $\lfloor \cdot \rfloor$, $E(enab(t_j, \mathbf{M}))$ may be different to $enab(t_j, \mathbf{m})$ in a non-ordinary net ($w > 1$). The similar problem may appear even in an ordinary net when t_j is a join ($|\bullet t_j| > 1$): $E(enab(t_j, \mathbf{M})) = E(\min\{\mathbf{M}[p_i]\}, p_i \in \bullet t_j)$ is not equal to

2.3. Timed continuous Petri nets

$enab(t_j, \mathbf{m}) = \min\{\mathbf{m}[p_i]\} \sim \min\{E(\mathbf{M}[p_i])\}, p_i \in \bullet t_j$, 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 appear; 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 fluidization 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): $\mathbf{m}_{k+1} = \mathbf{m}_k + \mathbf{C}(\mathbf{\Lambda}\mathbf{\Pi}(\mathbf{m}_k)\mathbf{m}_k\Delta\tau + \mathbf{v}_k)$ where \mathbf{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 ($\mathbf{m}[p_i]^{q-1}/q^q$) to the flow of a transition t_j , such that $p_i = \bullet t_j$ and $\mathbf{Pre}[p_i, t_j] = q$ [87], then the flow of t_j is modified to $\mathbf{f}[t_j] = \lambda_j \cdot (\mathbf{m}[p_i]/q)^q$. 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 shown 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 \mathbf{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*).

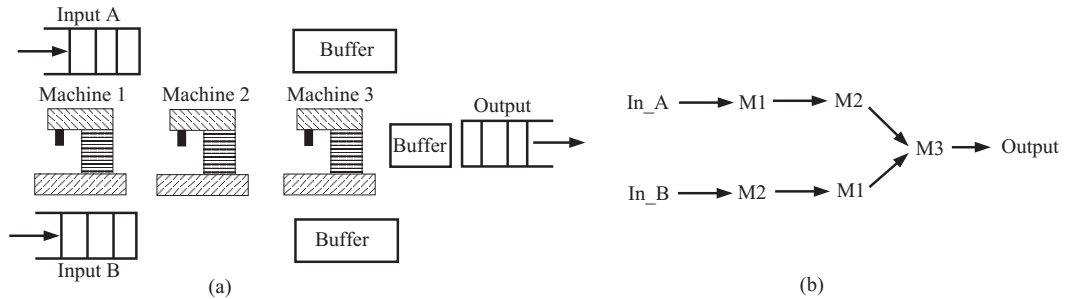


Figure 2.5: (a) Logical layout of a manufacturing system and (b), its production process

2.4. An example: a kanban-like manufacturing system

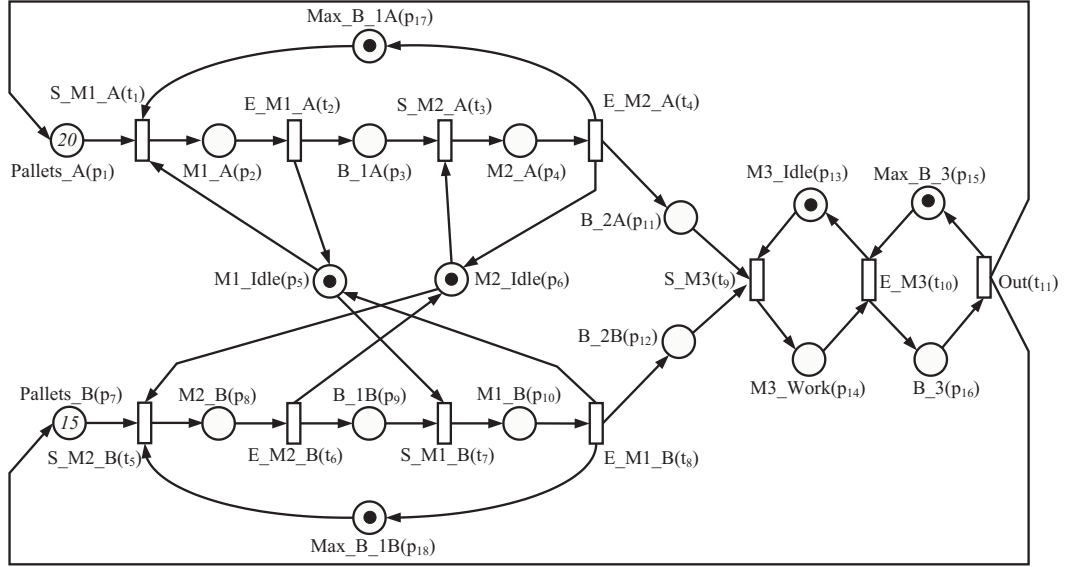


Figure 2.6: The PN system that models the manufacturing system described in Fig.2.5

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*: $\exists \mathbf{y} > 0$ s.t. $\mathbf{y}^T \cdot \mathbf{C} = 0$ (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 $\mathbf{x} = \mathbf{1}$ ($\mathbf{C} \cdot \mathbf{x} = 0$). 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
\mathbf{y}_1	$M1_A + M1_Idle + M1_B = 1$
\mathbf{y}_2	$Pallets_B + M2_B + B1_B + M1_B + B_2B + M3_work + B3 = 15$
\mathbf{y}_3	$M2_B + B1_B + M1_B + Max_B1_B = 1$
\mathbf{y}_4	$Pallets_A + M1_A + B1_A + M2_A + B_2A + M3_work + B3 = 20$
\mathbf{y}_5	$M1_A + B1_A + M2_A + Max_B1_A = 1$
\mathbf{y}_6	$M2_A + M2_Idle + M2_B = 1$
\mathbf{y}_7	$Max_B_3 + B_3 = 1$
\mathbf{y}_8	$M3_Idle + M3_work = 1$

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

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 $\mathbf{m}_0'[p_5] = 2$. 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 $\langle \mathcal{N}, \mathbf{M}_0 \rangle$ 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 δ , is set as following.

2.4. An example: a kanban-like manufacturing system

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

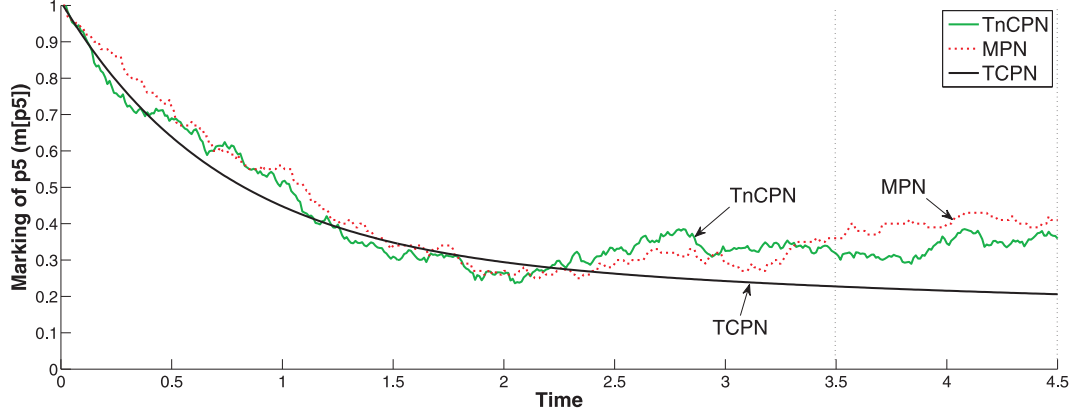
As an important part in the life-cycle of manufacturing systems, performance evaluation has been widely investigated by using time interpreted PNs, under both the framework of continuous and discrete systems, for example in [50, 69, 8]. We will focus on the *steady state* evaluation and *transient state* 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 $\mathbf{m}_{ss} = [0.1 \ 0.3 \ 0.1 \ 0.4 \ 0.2 \ 0.3 \ 13.5 \ 0.3 \ 0.1 \ 0.5 \ 18.6 \ 0.1 \ 0.6 \ 0.4 \ 0.9 \ 0.1 \ 0.2 \ 0.1]^T$.

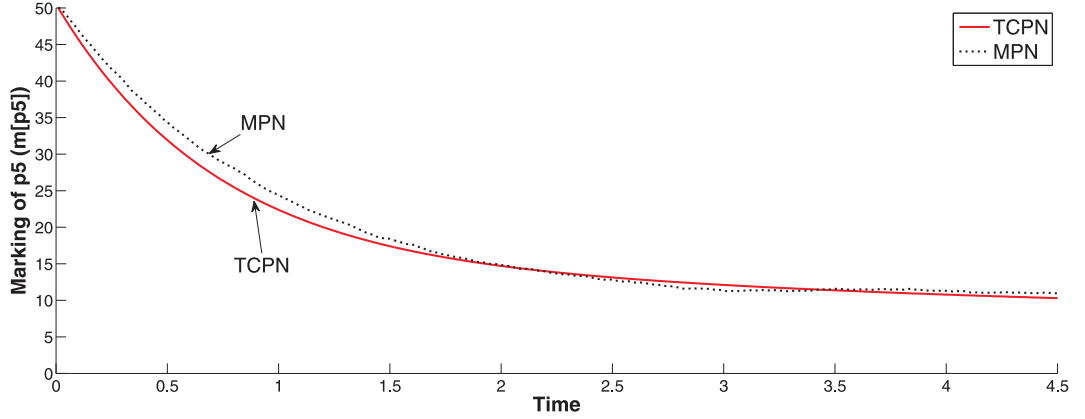
Transient analysis: The transient evaluation analyzes the behavior of the system, from the initial state (at time zero) until a given end time. As we have mentioned in the previous sections, TCPN models can approximate the average marking of the corresponding MPN if the MPN evolves inside a unique region (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 $\mathbf{M}[p_5]$ (for the MPN) is 0.40, while the average value of $\mathbf{m}[p_5]$ (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 $\mathbf{m}[p_5]$ 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 \mathbf{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 \mathbf{m}_0

limited to 1. In the case that the system is more populated, for example, instead of using \mathbf{m}_0 , we simulate the system with initial marking equal to $50 \cdot \mathbf{m}_0$ (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 $50 \cdot \mathbf{m}_0$

Last but not least, let us consider the parametric optimization problem of computing the optimal initial marking that satisfies a linear constraint $\mathbf{K} \cdot \mathbf{m}_0 \leq \mathbf{d}$. Assume that because of constraints on the investment, we can have at most 5 machines in the system ($\mathbf{m}_0[M1_Idle] + \mathbf{m}_0[M2_Idle] + \mathbf{m}_0[M3_Idle] \leq 5$), and the total size of buffers is constrained to no more than 10 ($\mathbf{m}_0[Max_B_1A] + \mathbf{m}_0[Max_B_1B] + \mathbf{m}_0[Max_B_3] \leq 10$). At the same time, the total amount of available pallets for the raw materials A and B are limited to 20 ($\mathbf{m}_0[Pallets_A] + \mathbf{m}_0[Pallets_B] \leq 20$). Under these constraints and with the other places initially set to be zero, we want to compute an optimal \mathbf{m}_0 , such that the throughput of transition *Out* (t_{11}) in

2.4. An example: a kanban-like manufacturing system

the steady state is maximized. An optimal \mathbf{m}_0 obtained by solving LPP (2.13) is: $\mathbf{m}_0[Pallets_A] = 3.4$, $\mathbf{m}_0[Pallets_B] = 3.6$, $\mathbf{m}_0[Max_B_3] = 1.1$, $\mathbf{m}_0[Max_B_1A] = 2.0$, $\mathbf{m}_0[Max_B_1B] = 2.3$, $\mathbf{m}_0[M1_Idle] = 2.1$, $\mathbf{m}_0[M2_Idle] = 1.8$, $\mathbf{m}_0[M1_Idle] = 1.1$. 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:

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

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

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

A transition t_j is said to be *uncontrollable* if the only control input that can be applied is $\mathbf{u}(\tau)[t_j] = 0$. The transitions set T can be partitioned into disjoint sets 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.

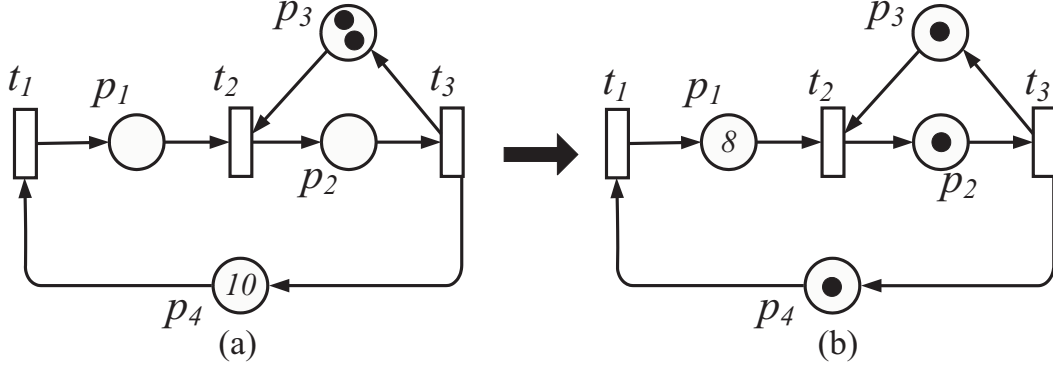
Among others, we will consider two related problems: (minimum-time) target marking control and (minimum-time) optimal flow control.

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 \mathbf{m}_f , from a given initial state \mathbf{m}_0 . In particular, we address the problem of reaching \mathbf{m}_f in minimum-time. Then, we are able to maintain \mathbf{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.

3.1. Introduction

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



Assume that we want to drive the system to $\mathbf{m}_f = [8 \ 1 \ 1]^T$ (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 ($\mathbf{u} = 0$), the system state will automatically evolve to \mathbf{m}_f and then \mathbf{m}_f is maintained, in around 6.1 time units. In order to reach \mathbf{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 \mathbf{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 $\mathbf{m}_f' = [1 \ 1 \ 1 \ 8]^T$. To reach \mathbf{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 $\mathbf{m}_f = [8 \ 1 \ 1 \ 1]^T$ in the previous example, i.e., fire $t_1(2)$ and block other transitions; then fire $t_2(1)$, the system state reaches \mathbf{m}_f' in only 0.91 time units. Therefore the maximal flow is achieved much faster (than in the case of reaching \mathbf{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 \mathbf{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 \mathbf{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 \mathbf{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 $\mathbf{m}_0 > 0$. 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 \mathbf{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 $\mathbf{m}_0 = [0 \ 0 \ 1]^T$ the system deadlocks. There exists no control law to reach a final state $\mathbf{m}_f = [0 \ 1 \ 0]^T$ even if \mathbf{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 $\mathbf{m}_0 \geq 0$. 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. Computing the initial and desired final states

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 \mathbf{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:

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

where $\mathbf{v}[p_i, t]$ are *slack* variables; \mathbf{m}_{ss} is a steady state marking, \mathbf{w}_{ss} is the (controlled) flow in the steady-state, and \mathbf{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 $\mathbf{v}[p_i, t_j]$, the original non-linear problem is transformed to a LPP. When p_i is the unique input place of t_j , variable $\mathbf{v}[p_i, t_j]$ can be viewed as the control input that reduces its flow. It is obvious that, if t_j is a synchronization, the minimal one, $\mathbf{u}[t_j] = \min_{p_i \in \bullet t_j} \mathbf{v}[p_i, t_j]$, 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 ($\mathbf{m}_f = \mathbf{m}_{ss}$) with the minimal WIP cost, $\mathbf{l} \cdot \mathbf{m}_{ss}$ (where \mathbf{l} is the work-in-process (WIP) cost vector), is computed by solving LPP:

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

- 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 $\mathbf{m}_0 > 0$, 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., $\mathbf{m}_f > 0$.

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. More generally speaking, it is related to the classical controllability concept, according to which *a system is controllable* if for any two states $\mathbf{m}_1, \mathbf{m}_2$ of the state space it is possible to transfer the system from \mathbf{m}_1 to \mathbf{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., $\mathbf{0} \leq \mathbf{u} \leq \Lambda \Pi(\mathbf{m})\mathbf{m}$, 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 \mathbf{y} is a *P-flow* then any reachable marking \mathbf{m} must fulfill $\mathbf{y}^T \mathbf{m} = \mathbf{y}^T \mathbf{m}_0$, defining thus a *state invariant*. Nevertheless, the study of controllability “over” this invariant is particularly interesting. This set is formally defined as $Class(\mathbf{m}_0) = \{\mathbf{m} \in \mathbb{R}_{\geq 0}^{|P|} | \mathbf{B}_y^T \mathbf{m} = \mathbf{B}_y^T \mathbf{m}_0\}$, where \mathbf{B}_y is a basis of P-flows, i.e., $\mathbf{B}_y^T \mathbf{C} = \mathbf{0}$. For a general TCPN system, every reachable marking belongs to $Class(\mathbf{m}_0)$ (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 $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ is controllable with bounded input (BIC) over $S \subseteq Class(\mathbf{m}_0)$ if for any two markings $\mathbf{m}_1, \mathbf{m}_2 \in S$ there exists an input \mathbf{u} transferring the system from \mathbf{m}_1 to \mathbf{m}_2 in finite or infinite time, and it is suitably bounded, i.e., $\mathbf{0} \leq \mathbf{u} \leq \Lambda \Pi(\mathbf{m})\mathbf{m}$, and $\forall t_i \in T_{nc} \mathbf{u}[t_i] = 0$ along the marking trajectory.*

3.3. Controllability

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

Example 3.3.3. Consider for instance the TCPN of Fig. 3.2(a) and the markings $\mathbf{m}_0 = [2 \ 1 \ 1]^T$, $\mathbf{m}_1 = [1 \ 1 \ 2]^T$ and $\mathbf{m}_2 = [1 \ 2.5 \ 0.5]^T$. Obviously, \mathbf{m}_1 and \mathbf{m}_2 are both in $\text{Class}(\mathbf{m}_0)$. 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 $\exists \sigma \geq \mathbf{0}$ such that $\mathbf{m}_1 = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$, but $\nexists \sigma \geq \mathbf{0}$ such that $\mathbf{m}_2 = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$. So \mathbf{m}_1 is reachable but \mathbf{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 $\text{Class}(\mathbf{m}_0)$. The shadowed area in Fig.3.2(a) corresponds to the set of reachable markings. It is the convex cone defined by vectors \mathbf{c}_1 , \mathbf{c}_2 and \mathbf{c}_3 , which represent the columns of \mathbf{C} (here restricted to p_1 and p_3).

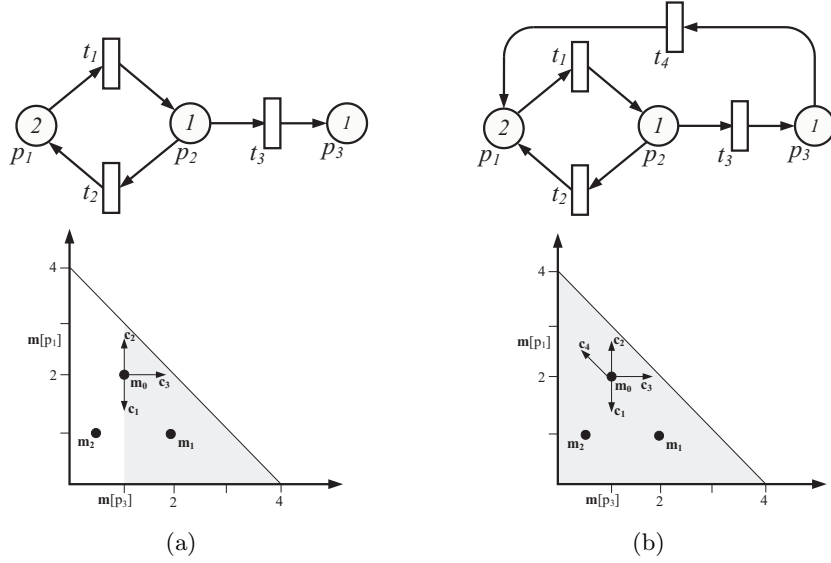


Figure 3.2: Two TCPN systems with identical P-flows and initial marking. The shadowed areas correspond to the sets of reachable markings. The net in (b) is consistent and there exists no emptied siphon, therefore controllable over $\text{Class}(\mathbf{m}_0)$.

Now, consider the system of Fig. 3.2(b). In this case, \mathbf{m}_2 become reachable from \mathbf{m}_0 . In fact, for any marking $\mathbf{m} \in \text{Class}(\mathbf{m}_0)$, the vector $(\mathbf{m} - \mathbf{m}_0)$ is in the convex cone defined by the vectors \mathbf{c}_1 to \mathbf{c}_4 , which occurs due to the consistency of the net and implies that \mathbf{m} is reachable from \mathbf{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 $\text{Class}(\mathbf{m}_0)$.

If uncontrollable transitions exist, the analysis of controllability becomes more complex, and in general the systems are no longer controllable over $Class(\mathbf{m}_0)$, 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 $\mathbf{m}^q \in Class(\mathbf{m}_0)$ is an equilibrium one if $\exists \mathbf{u}^q$ ($0 \leq \mathbf{u}^q \leq \Lambda \Pi(\mathbf{m}^q) \mathbf{m}^q$) such that $\mathbf{C}(\Lambda \Pi(\mathbf{m}^q) \mathbf{m}^q - \mathbf{u}^q) = \mathbf{0}$. 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 $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ be a TCPN system. Consider some equilibrium sets S_1, S_2, \dots, S_j related to different regions $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_j$. 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:

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

where $p_{j1}, p_{j2}, \dots, p_{jn}$ 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

3.4. Previous centralized control methods

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 \mathbf{m}_0 and \mathbf{m}_d , and the markings in the intersection of l with the region’s borders, denoted as $\mathbf{m}_c^1, \mathbf{m}_c^2, \dots, \mathbf{m}_c^n$. Define $\mathbf{m}_c^0 = \mathbf{m}_0$ and $\mathbf{m}_c^{n+1} = \mathbf{m}_f$. Then, $\forall k \in \{0, 1, \dots, n\}$ compute τ_k by solving the linear programming problem (LPP):

$$\begin{aligned} \min \quad & \tau_k \\ \text{s.t. : } \quad & \mathbf{m}_c^{k+1} = \mathbf{m}_c^k + \mathbf{C} \cdot \mathbf{x} \\ & 0 \leq \mathbf{x}_j \leq \lambda_j \Pi_{ji}^k \min\{\mathbf{m}_c^k[p_i], \mathbf{m}_c^{k+1}[p_i]\} \tau_k \\ & \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji}^k \neq 0 \end{aligned} \quad (3.3)$$

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

The control law to be applied is thus $\mathbf{w} = \mathbf{x}/\tau_k$, when the system is between the markings \mathbf{m}_c^k and \mathbf{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 \mathbf{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.

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

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: $\mathbf{m}_{k+1} = \mathbf{m}_k + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_k$, subject to the constraints $\mathbf{0} \leq \mathbf{w}_k \leq \mathbf{f}_k$ with \mathbf{f}_k being the flow without control, which is equivalent to $\mathbf{G} \cdot [\mathbf{w}_k^T, \mathbf{m}_k^T]^T \leq \mathbf{0}$, for a particular matrix \mathbf{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:

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

$$\mathbf{G} \cdot \begin{bmatrix} \mathbf{w}_{k+j} \\ \mathbf{m}_{k+j} \end{bmatrix} \leq \mathbf{0}, j = 0, \dots, N-1 \quad (3.5b)$$

$$\mathbf{w}_{k+j} \geq \mathbf{0}, j = 0, \dots, N-1 \quad (3.5c)$$

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

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

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

However, if the desired marking (\mathbf{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 \mathbf{m}_k to \mathbf{m}_f . Roughly, this is equivalent to add a terminal constraint in the form of:

3.4. Previous centralized control methods

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

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 $\mathbf{w}(k) = \mathbf{F}_r(\mathbf{m}(k) - \mathbf{m}_d) + \mathbf{R}$, where \mathbf{R} is a vector and \mathbf{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: $\mathbf{0} \leq \mathbf{w}(k) \leq \mathbf{f}(k)$, where $\mathbf{f}(k)$ represents the flow without control,
2. the region membership: $\mathbf{m}(k) \in \mathcal{P}(\mathbf{G}_r, \mathbf{g}_r)$, where $\mathcal{P}(\mathbf{G}_r, \mathbf{g}_r) = \{\mathbf{m} | \mathbf{G}_r \mathbf{m} \leq \mathbf{g}_r\}$ 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: $\mathbf{x}(k) \in \mathcal{P}(\mathbf{Q}, \boldsymbol{\mu}) \rightarrow \mathbf{x}(k+1) \in \mathcal{P}(\mathbf{Q}, \alpha \boldsymbol{\mu})$, where $\mathbf{x}(k) = (\mathbf{m}(k) - \mathbf{m}_d)$ is the current error, $\alpha < 1$ and $\mathcal{P}(\mathbf{Q}, \alpha \boldsymbol{\mu}) = \{\mathbf{x} | \mathbf{Q} \mathbf{x} \leq \alpha \boldsymbol{\mu}\}$ is the contractive set (so, the absolute error is monotonically decreasing).

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 \mathbf{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:

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

with restriction $\mathbf{x} \in \chi$ and $\mathbf{u} \in U$, where U is a polytope of admissible inputs. An admissible *affine control law* is a affine function $\mathbf{u} : \chi \rightarrow U$, characterized by $\mathbf{u}(\mathbf{x}) = \mathbf{F}\mathbf{x} + \mathbf{g}$. 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 assumed 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 $\mathbf{m}_0 > 0$, using this affine control technology, the system can always be driven to a desirable final state $\mathbf{m}_f > 0$. 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; another 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 $\mathbf{m}_0 > 0$, $\mathbf{m}_f > 0$ 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

3.4. Previous centralized control methods

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 $\mathbf{m}_{01} = [3 \ 3 \ 1 \ 3]^T$; $\mathbf{m}_{02} = [2.1 \ 2.1 \ 0.1 \ 2.1]^T$ and also two different final marking: $\mathbf{m}_{f1} = [1 \ 4.5 \ 1.5 \ 3]^T$ and $\mathbf{m}_{f2} = [0.1 \ 3.6 \ 0.6 \ 2.1]^T$, respectively. It can be checked that \mathbf{m}_{f_i} can be reached from \mathbf{m}_{0i} , $i = 1, 2$ with the same firing count vectors, but one element of \mathbf{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 $\mathbf{m}_{01}/\mathbf{m}_{f1}$) and Table 3.3 (considering $\mathbf{m}_{02}/\mathbf{m}_{f2}$).

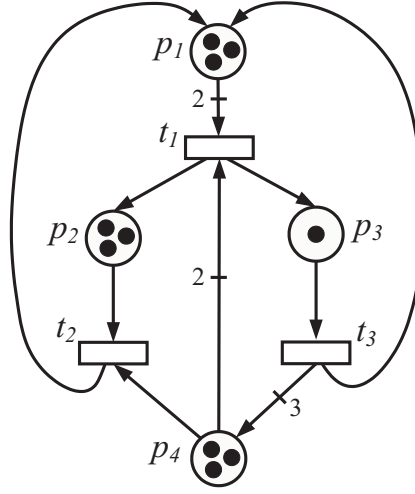


Figure 3.3: A simple TCPN system

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

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 \mathbf{m}_{f_1} from \mathbf{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 \mathbf{m}_{f_2} from \mathbf{m}_{02} (Table 3.3), the approaching minimum-time controller does not work very well. Because in this approach, between each pair

3.4. Previous centralized control methods

Table 3.3: Simulation results of reaching \mathbf{m}_{f_2} from \mathbf{m}_{0_2} , in the net system of Fig. 3.3 (for the MPC control, the weight matrix $\mathbf{Q} = q \cdot \mathbf{1}^{|P|}$, $\mathbf{R} = r \cdot \mathbf{1}^{|T|}$)

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 \mathbf{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 \mathbf{Q} and a small weight for \mathbf{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)), $\mathbf{m}_f = [18.78 \ 0.39 \ 0.13 \ 0.52 \ 0.16 \ 0.29 \ 13.65 \ 0.39 \ 0.13 \ 0.65 \ 0.13 \ 0.13 \ 0.58 \ 0.52 \ 0.97 \ 0.13 \ 0.26 \ 0.13]^T$. 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 $\mathbf{Q} = q \cdot \mathbf{1}^{|P|}$, $\mathbf{R} = r \cdot \mathbf{1}^{|T|}$)

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 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 from 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.

3.5. Conclusions

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 been 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]:

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

where $\mathbf{u}(\delta)$, $\mathbf{w}(\delta)$ 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 braking 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., $\mathbf{m}_0 > 0$) is not mandatory; while for the extended controllers of general nets, we assume $\mathbf{m}_0 > 0$. We always assume a positive final state (i.e., $\mathbf{m}_f > 0$), 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 \mathbf{m} can be reached from \mathbf{m}_0 by using different firing sequences. For example, if the net is consistent and \mathbf{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 $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ be a CPN system and \mathbf{m}_f be a reachable marking through a sequence σ , i.e., $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$. A firing count vector σ is said to be minimal if for any T-semiflow \mathbf{x} , $\|\mathbf{x}\| \not\subseteq \|\sigma\|$, where $\|\cdot\|$ stands for the support of a vector. We can simply compute a σ by solving the following LPP:

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

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 $\mathbf{m}_0 = [4 \ 0 \ 0 \ 0 \ 0]^T$ and $\mathbf{m}_f = [2 \ 0 \ 0 \ 0 \ 1]^T$. To drive the system to its final state, there exist two minimal firing count vectors $\sigma_1 = [1 \ 1 \ 0 \ 1 \ 0]^T$ and $\sigma_2 = [0 \ 0 \ 1 \ 0 \ 1]^T$. The final state can also be reached by firing $\sigma_3 = [1 \ 1 \ 1 \ 1 \ 1]^T$, but σ_3 is not a minimal firing count vector, because it contains a T-semiflow $[0 \ 0 \ 1 \ 0 \ 1]^T$.

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

- (A1) The matrix \mathbf{C} has full rank;
- (A2) The net is strongly connected and consistent.

Proof: Suppose there exist two minimal firing count vectors σ_1 and σ_2 , then (1) $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \sigma_1$, (2) $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \sigma_2$. Subtracting (2) from (1), we obtain:

$$\mathbf{C} \cdot (\sigma_1 - \sigma_2) = \mathbf{C} \cdot \sigma_{12} = 0$$

If (A1) holds, we must have $\sigma_{12} = 0$, so $\sigma_1 = \sigma_2 (\neq 0, \text{ if } \mathbf{m}_f \neq \mathbf{m}_0)$.

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

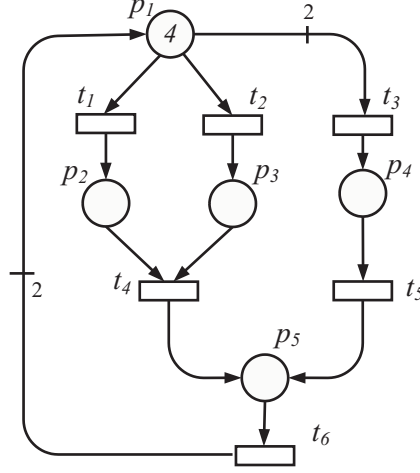


Figure 4.1: A non-CF Petri net system

0. Since $\mathbf{C} \cdot (\boldsymbol{\sigma}_{12} + \alpha \cdot \mathbf{x}) = 0$ and $\boldsymbol{\sigma}_{12} + \alpha \cdot \mathbf{x} \geq 0$, it is a T-semiflow. Therefore, there exists $\beta > 0$ such that $\boldsymbol{\sigma}_{12} + \alpha \cdot \mathbf{x} = \beta \cdot \mathbf{x}$, implying $\boldsymbol{\sigma}_{12} = (\beta - \alpha) \cdot \mathbf{x}$. If $\beta - \alpha = 0$ then $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2$ which is impossible by assumption. If $\beta - \alpha > 0$ then $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 + (\beta - \alpha) \cdot \mathbf{x} > (\beta - \alpha) \cdot \mathbf{x}$. 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 \mathbf{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*, \mathbf{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:

$$\begin{aligned} \mathbf{m}_{k+1} &= \mathbf{m}_k + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_k \\ 0 &\leq \mathbf{w}_k \leq \mathbf{f}_k \end{aligned} \quad (4.3)$$

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

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

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.

4.2. Minimum-time controller for Choice-Free nets

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 $\mathbf{m}_f = [0.2 \ 0.5 \ 0.3]^T$, 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 \mathbf{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.

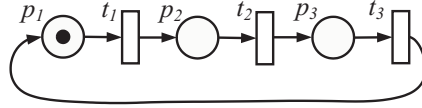


Figure 4.2: A trivial CF net system with $\mathbf{m}_0 = [1 \ 0 \ 0]$.

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 $\boldsymbol{\sigma}[t_j]$ ($\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$) is reached. The control input of t_j at k^{th} sampling period is:

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

where at $k = 0$, $\Theta \cdot \sum_{i=0}^{k-1} \mathbf{w}_i[t_j] = 0$. Remember $\mathbf{w}_k = \mathbf{f}_k - \mathbf{u}_k$, (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. $\mathbf{u}_k[t_j] = 0$; (b) means once $\boldsymbol{\sigma}[t_j]$ is reached, the transition is completely stopped (OFF), i.e. $\mathbf{u}_k[t_j] = \mathbf{f}_k[t_j]$; (c) represents the last firing of t_j . Algorithm 1 synthesizes the ON/OFF controller, in which $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots$ is the sequence of control inputs at the time instants.

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

Algorithm 1 ON/OFF controller

Input: $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle, \mathbf{m}_f, \sigma, \Theta$
Output: w_0, w_1, w_2, \dots

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

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

- 4: Apply $\mathbf{w}_k : \mathbf{m}_{k+1} = \mathbf{m}_k + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_k$
 - 5: $k := k + 1$
 - 6: **end while**
 - 7: **return** w_0, w_1, w_2, \dots
-

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

Proof: Assume an arbitrary non ON/OFF controller \mathbf{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 \mathbf{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 σ , 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 σ .)

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

4.2. Minimum-time controller for Choice-Free nets

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

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

Since by firing σ' and σ the same final state is reached, $\sigma'[t_j] \geq \sigma[t_j]$ implies that in the case of firing σ' , more tokens should be put into each of its input place $p_i \in {}^\bullet t_j$ (with the quantity of $\text{Pre}[p_i, t_j] \cdot (\sigma'[t_j] - \sigma[t_j])$), 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 $\sigma[t_j]$ (they do not make $\sigma[t_j]$ firing faster). Therefore, firing $\sigma'[t_j]$ cannot cost less time than firing $\sigma[t_j]$. ■

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 $\lambda = [1 \ 0.01 \ 1 \ 1 \ 1]^T$, the sampling period is $\Theta = 0.1$, and we want to reach a final state $\mathbf{m}_f = [0 \ 0.5 \ 0.5 \ 4]^T$. \mathbf{m}_f can be reached, for example, by using $\sigma = [0 \ 0.5 \ 0 \ 0 \ 0]^T$ or $\sigma' = [0 \ 0.5 \ 0 \ 4 \ 4]^T$. Although $\sigma \leq \sigma'$, we can verify that if we apply the ON/OFF controller using σ the final state is reached in 692 time steps; if we we apply the ON/OFF controller using σ' the final state is reached in only 673 time steps. In the case of σ , only t_2 fires, so the time used for the firing of $\sigma[t_2]$ determines the time to reach the final state. In the case of σ' , 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 $\sigma'[t_2]$ still determines the total time to reach \mathbf{m}_f . Therefore, the final state is reached faster in the case of firing σ' .

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

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 $\mathbf{m}_0 = [1 \ 1 \ 0]^T$; $\lambda = [10 \ 1 \ 1]^T$; and sampling period is $\Theta = 0.01$. Given $\sigma = [0 \ 0.5 \ 0]^T$ and $\sigma' = [0.5 \ 0.5 \ 0]^T$ ($\sigma \leq \sigma'$), by using the ON/OFF controller, σ is fired in 69 time steps ($[1 \ 0.5 \ 0.5]^T$ is reached) and σ' is fired in only 42 time steps ($[0.5 \ 1 \ 0.5]^T$ is reached).

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

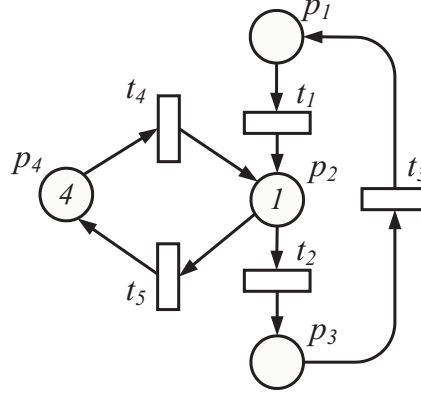


Figure 4.3: A simple non-CF net (a state machine): by using the ON/OFF controller, firing σ may cost more time than firing $\sigma' \geq \sigma$

to $\mathbf{m}_f > 0$, i.e., $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$. The ON/OFF controller is a minimum-time controller driving the system to \mathbf{m}_f if σ 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 $\mathbf{x} > 0$ [93], hence for any other firing count vector $\sigma' \neq \sigma$ that drives the system to \mathbf{m}_f we must have $\sigma' = \sigma + \alpha \cdot \mathbf{x} > \sigma$, $\alpha > 0$. According to Lemma 4.2.5 and 4.2.6, the results can be derived straightforwardly. ■

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

4.2.3 ON/OFF controller: continuous-time case

By taking the sampling period $\Theta \rightarrow 0$, 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:

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

where σ is the minimal firing count vector and $\mathbf{w}(\delta)[t_j]$ is the controlled flow of t_j at time δ ; $\mathbf{f}(\tau)[t_j]$ 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 \mathbf{m}_f , the final marking is reached at the limit, i.e., in infinite time. If $\mathbf{m}_f > 0$ and use the proposed control method, this situation does not happen.

4.2. Minimum-time controller for Choice-Free nets

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) at 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 \mathbf{m}_f in general.

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

Proof: We prove it by contradiction. Assume that the system reaches a state \mathbf{m} and $\mathbf{m} \neq \mathbf{m}_f$. Therefore, there must exist a transition t_j that cannot reach its accumulative firing upper bound $\sigma[t_j]$. Since the system is JF, t_j can fire if its unique input place p_i is not emptied, it implies that $\mathbf{m}[p_i] = 0$. For any other place p'_i such that its output transitions fire completely the firing amounts give by σ , it holds $\mathbf{m}[p'_i] \leq \mathbf{m}_f[p'_i]$ because the firing of every transition is upper bounded by σ and $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$. Therefore, for any place $p \in P$ it holds $\mathbf{m}[p] \leq \mathbf{m}_f[p]$ and there exists at least one place $p_i \in P$, such that $\mathbf{m}[p_i] = 0 < \mathbf{m}_f[p_i]$. 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 $\sigma = [0.85 \ 0.85 \ 1.0 \ 0.9 \ 0.6 \ 0.6 \ 0.75 \ 0.65 \ 0.45 \ 0.2 \ 0.35 \ 0.10]^T$.

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.

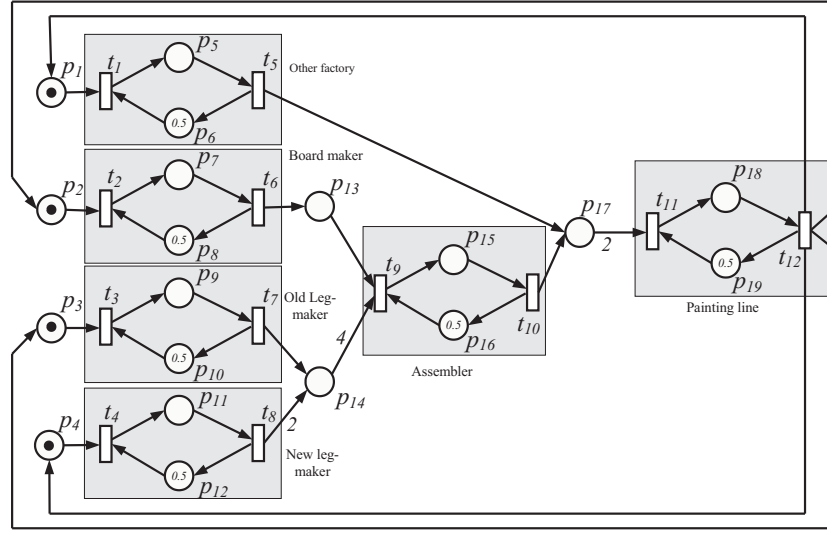


Figure 4.4: The CF net model (weighted T-system) of a table factory system. The firing rate of every transition is equal to 1.

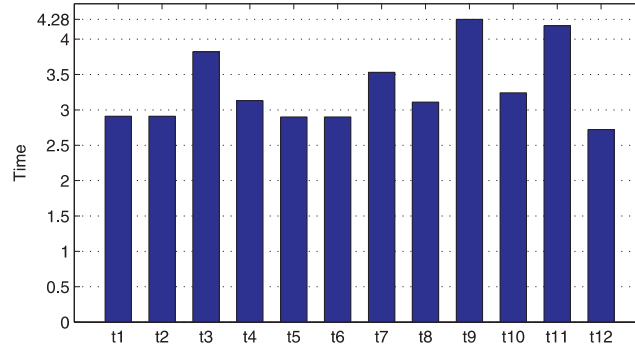


Figure 4.5: Transitions' stopping time instants obtained by applying the ON/OFF controller to the CF system in Fig. 4.4

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.

4.3. Drawbacks of the ON/OFF controller for general nets

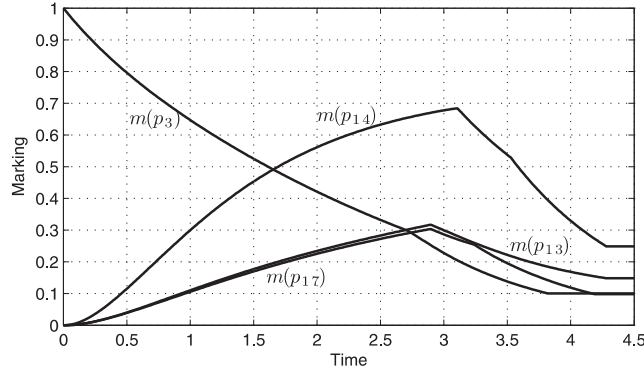


Figure 4.6: Marking trajectories of applying the ON/OFF controller to the CF system in Fig. 4.4

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

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 $\sigma[t_7] = 0$), bringing the system to some “blocking” situations.

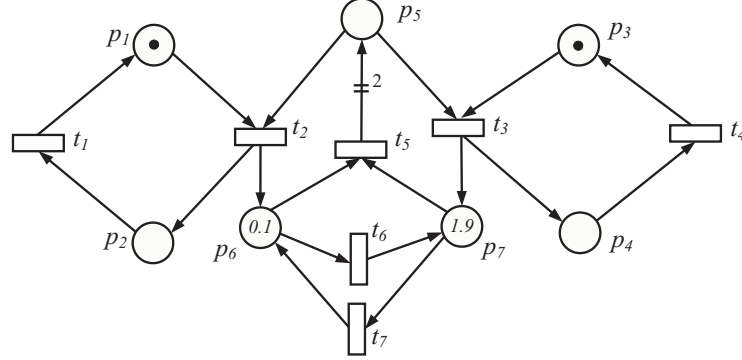


Figure 4.7: A live and bounded CPN system that the ON/OFF controller brings to a “deadlock” situation if $\lambda_3 \gg \lambda_2$

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

4.4. Extended ON/OFF based methods

Algorithm 2 ON/OFF+ controller

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

Output: w_0, w_1, w_2, \dots

- 1: $k \leftarrow 0$
- 2: **while** $\Theta \cdot \sum_{i=0}^{k-1} w_i \neq \sigma$ **do**
- 3: Solve the following LPP:

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

- 4: Apply $\mathbf{w}_k : \mathbf{m}_{k+1} \leftarrow \mathbf{m}_k + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_k$
 - 5: $k \leftarrow k + 1$
 - 6: **end while**
 - 7: **return** w_0, w_1, w_2, \dots
-

means that, at any time step k , if transitions t_a and t_b are in conflict, the following will be forced: $\frac{w_k[t_a]}{w_k[t_b]} = \frac{\sigma[t_a]}{\sigma[t_b]}$. Since we need positive flows ($w_k[t_b] > 0$), in the sequel we assume an initial state $\mathbf{m}_0 > \mathbf{0}$. 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 \mathbf{m}_f , implying that the ON/OFF+ controller also drives the original one to \mathbf{m}_f .

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

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

Example 4.4.1. Let $m > 0$ and $\sigma[t_1] > 0$, $\sigma[t_2] > 0$. Fig. 4.8 shows how two conflicting transitions t_1 and t_2 are merged into $t_{1,2}$.

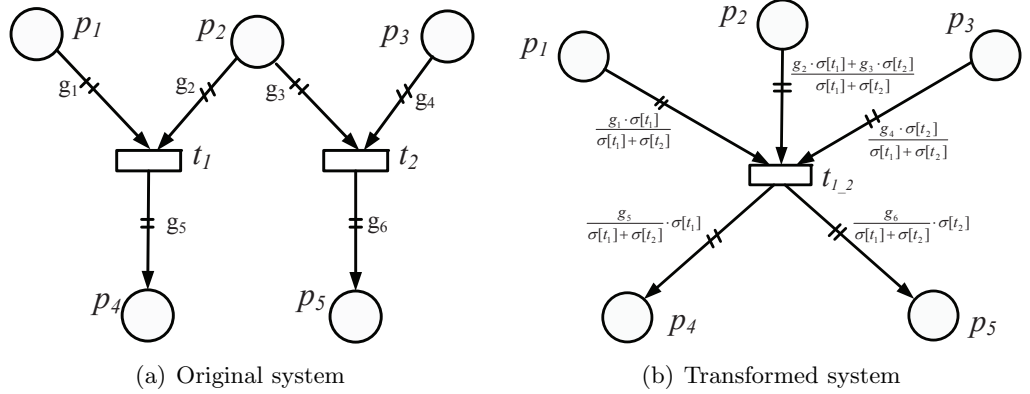


Figure 4.8: Dynamic reduction rule for a given σ : merging t_1 and t_2

Proposition 4.4.2. Let $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$, and $\mathcal{S}' = \langle \mathcal{N}', \mathbf{m}_0 \rangle$ be the transformed system from \mathcal{S} by merging $T_j = \{t_1, t_2, \dots, t_n\}$ 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 σ , 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 $s_1 = \alpha \cdot \sigma[t_1]$, $s_2 = \alpha \cdot \sigma[t_2]$, $\alpha > 0$. If $t_1(s_1)t_2(s_2)$ is fired in the original system, the new marking of p_2 is:

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

In the transformed system, if $t_{1,2}(s_1 + s_2)$ is fired, the new making of p_2 is:

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

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

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

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

4.4. Extended ON/OFF based methods

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

Proof: Let $\mathcal{S}' = \langle \mathcal{N}', \lambda', \mathbf{m}_0 \rangle$ 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 $\mathbf{w}'_k[t_j]$ 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 $\mathbf{w}'_k[t_j] = \sum_{t \in T_j} \mathbf{w}_k[t]$; (2) otherwise $\mathbf{w}'_k[t_j] = \mathbf{w}_k[t_j]$, where $\mathbf{w}_k[t_j]$ 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}' , \mathbf{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 $\mathbf{u}_k[t_j]$. In particular, it is defined as:

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

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

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

where $\text{enab}(t_d, \mathbf{m}_k)$, $t_d \in T_j$, is the enabling degree of t_d in the original system at \mathbf{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 σ , if transition t_j is a persistent one and the goal is to minimize the time spent for firing $\sigma[t_j]$, 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 $\mathbf{m}_f = [3.6 \ 0.4 \ 4 \ 1.6]^T$, the firing rate vector $\lambda = \mathbf{1}$, and the sampling*

period $\Theta = 0.2$. One minimal firing count vector to reach \mathbf{m}_f (in this case, the unique one) is $\sigma = [0.4 \ 0 \ 4 \ 0]^T$. 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, \mathbf{m}_f is reached in 12 time steps. However, \mathbf{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.

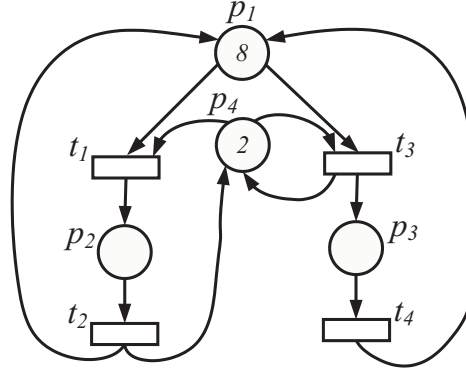


Figure 4.9: An EQ PN system with $\mathbf{m}_0 = [8 \ 0 \ 0 \ 2]^T$.

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 $\mathbf{m}_f > 0$. 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 σ is determined by the “slower” 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 σ 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 σ , 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

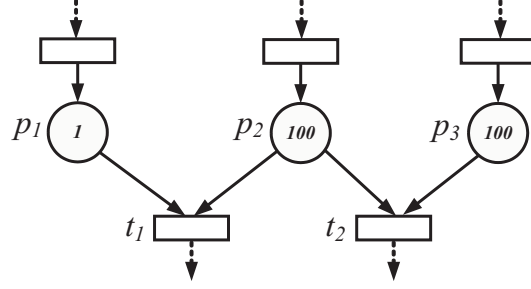


Figure 4.10: Fast transitions may be slowed down

“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 \mathbf{m} with \mathbf{w} its controlled flow, and let $\boldsymbol{\sigma}$ be the firing count vector that should be fired to reach \mathbf{m}_f . Then $s_j = \lceil \frac{\sigma[t_j]}{\Theta \cdot \mathbf{w}[t_j]} \rceil$ can be viewed as an estimation of the number of time steps that transition t_j needs to fire, assuming that t_j fires with a constant speed equal to $\mathbf{w}[t_j]$. 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 \mathbf{m}_0 is defined by:

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

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

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

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 \mathbf{m}_k , and the firing count vector $\boldsymbol{\sigma}'$ has been fired, i.e., $\mathbf{m}_k = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}'$. 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 \mathbf{m}_k is defined by:

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

where $\mathbf{w}_k[t_j]$ 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 $T_c^i, i = 1, 2, 3, \dots, l$ 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.

$$(C1) \quad D_b^k \leq d$$

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

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

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

4.4. Extended ON/OFF based methods

Algorithm 3 B-ON/OFF Controller

Input: $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle, \mathbf{m}_f, \sigma, \Theta, d, T_p, T_c^i, i = 1, 2, 3, \dots, l$

Output: w_0, w_1, w_2, \dots

```

1: Partition every  $T_c^i$  into  $T_1^i$  and  $T_2^i, i = 1, 2, \dots, l$ 
2:  $k \leftarrow 0$ 
3: while  $\Theta \cdot \sum_{i=0}^{k-1} w_i \neq \sigma$  do
4:   Obtain  $w_k[t_j]$  for any  $t_j \in T_p$  : applying the ON/OFF strategy
5:   for  $i = 1$  to  $l$  do
6:     For any  $t_j \in T_2^i$ :  $w_k[t_j] \leftarrow 0$ 
7:     Obtain  $w_k[t_j]$  for any  $t_j \in T_1^i$  : applying the ON/OFF+ strategy
8:   end for
9:   Apply  $w_k$  :  $\mathbf{m}_{k+1} \leftarrow \mathbf{m}_k + \Theta \cdot C \cdot w_k$ 
10:   $\sigma_{k+1} \leftarrow \sigma - \Theta \cdot \sum_{i=0}^k w_i$ 
11:  for  $i = 1$  to  $l$  do
12:    if  $T_2^i \neq \emptyset$  then
13:      Compute  $w_{k+1}[t_a], t_a \in T_1^i$ 
14:       $h^{k+1} \leftarrow \sigma_{k+1}[t_a] / (\Theta \cdot w_{k+1}[t_a])$ 
15:      for each  $t_b \in T_2^i$  do
16:         $s_b^{k+1} \leftarrow \sigma_{k+1}[t_b] / (\Theta \cdot \lambda_b \cdot \text{enab}(t_b, \mathbf{m}_{k+1}))$ 
17:         $D_b^{k+1} \leftarrow s_b^{k+1} / h^{k+1}$ 
18:        if  $D_b^{k+1} \leq d$  or  $D_b^{k+1} \geq D_b^k$  then
19:           $T_1^i \leftarrow T_1^i \cup \{t_b\}$ 
20:           $T_2^i \leftarrow T_2^i \setminus \{t_b\}$ 
21:        end if
22:      end for
23:    end if
24:  end for
25:   $k \leftarrow k + 1$ 
26: end while
27: return  $w_0, w_1, w_2, \dots$ 

```

Since the accumulative firing counts of transitions are upper bounded by σ , then we have $\mathbf{m} = \mathbf{m}_0 + C \cdot \sigma', 0 \leq \sigma' \leq \sigma$. Because $\mathbf{m}_0 > 0$, in a finite time $\mathbf{m} > 0$. Since $\sigma - \sigma' \geq 0$ and $\mathbf{m}_f = \mathbf{m} + C \cdot (\sigma - \sigma') > 0$, \mathbf{m}_f is reachable from \mathbf{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 $J(\mathbf{m}_k, N)$ (3.6):

MPC is usually used for optimizing trajectories subject to certain constraints. In our problem, the aim is to reach \mathbf{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: $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle, \mathbf{m}_f, \sigma, \Theta, \mathbf{Q}, \mathbf{R}, N, \epsilon, \zeta$

Output: $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots$

```

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

```

4.4. Extended ON/OFF based methods

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

$$\begin{aligned} \min \quad & J(\mathbf{m}_k, N) \\ \text{s.t. :} \quad & \mathbf{m}_{k+j+1} = \mathbf{m}_{k+j} + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_{k+j}, j = 0, \dots, N-1 \end{aligned} \quad (4.13a)$$

$$\mathbf{G} \cdot \begin{bmatrix} \mathbf{w}_{k+j} \\ \mathbf{m}_{k+j} \end{bmatrix} \leq 0, j = 0, \dots, N-1 \quad (4.13b)$$

$$\mathbf{w}_{k+j} \geq 0, j = 0, \dots, N-1 \quad (4.13c)$$

$$\Theta \cdot \sum_{j=0}^{N-1} \mathbf{w}_{k+j} \leq \boldsymbol{\sigma}_k \quad (4.13d)$$

$$\mathbf{m}_{k+1} \geq \mathbf{1} \cdot \epsilon \quad (4.13e)$$

$$\mathbf{1}^T \cdot \mathbf{w}_k \geq \zeta \quad (4.13f)$$

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

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

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

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

We define the cost function as:

$$J(\mathbf{m}_k, N) = \sum_{j=0}^N [(\mathbf{m}_{k+j} - \mathbf{m}_f)' \cdot \mathbf{Q} \cdot (\mathbf{m}_{k+j} - \mathbf{m}_f)] - \mathbf{w}_k' \cdot \mathbf{R} \cdot \mathbf{w}_k \quad (4.14)$$

By means of the item $-\mathbf{w}_k' \cdot \mathbf{R} \cdot \mathbf{w}_k$ in the cost function and choosing large values for $\mathbf{R}[j, j], t_j \in T_p$, 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 $\langle \mathcal{N}, \boldsymbol{\lambda}, \mathbf{m}_0 \rangle$ be a TCPN system with $\mathbf{m}_0 > 0$. Let $\mathbf{m}_f > 0$ be a reachable final marking, such that $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$. 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 $V(\mathbf{m}_k) = \mathbf{1}^T \cdot (\boldsymbol{\sigma} - \Theta \cdot \sum_{i=0}^k \mathbf{w}_i)$, where \mathbf{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 σ . Therefore, $V(\mathbf{m}_k) \geq \mathbf{0}$ and $V(\mathbf{m}_k) \neq 0$ until $\sigma = \Theta \cdot \sum_{i=0}^k \mathbf{w}_i$, i.e., until $\mathbf{m}_k = \mathbf{m}_f$. Now we need to prove that $V(\mathbf{m}_k)$ is strictly decreasing, and it is equivalent to prove that $\mathbf{w}_k \neq 0$ until σ is reached. Considering the last constraint (4.13f), we only need to prove that problem (4.13) is feasible until \mathbf{m}_f is reached.

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

$$\mathbf{m}_f = \mathbf{m}_k + \mathbf{C} \cdot (\sigma - \sigma') > 0 \quad (4.15)$$

so \mathbf{m}_f is reachable from $\mathbf{m}_k > 0$ [49]. Since the net is consistent, the marking of the system is able to move from \mathbf{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 \mathbf{m}_{k+1} is, for example, on the straight line from \mathbf{m}_k to \mathbf{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 $\mathbf{m}_0 > 0$, $\mathbf{m}_f > 0$).

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 at each time step	Min. time
ON-OFF+	All	a LPP at each time step	Heuristic Min. Time
B-ON/OFF	All	a LPP at each time step	Heuristic Min. Time
MPC-ON/OFF	All	a QPP at 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, at each time step only a LPP

4.4. Extended ON/OFF based methods

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.

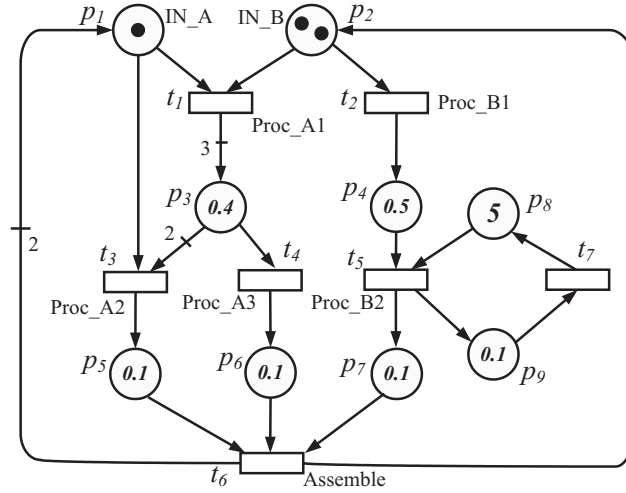


Figure 4.11: The TCPN model of an assembly system.

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) $\Theta = 0.01$, $\mathbf{m}_0 = [1 \ 2 \ 0.4 \ 0.5 \ 0.1 \ 0.1 \ 0.1 \ 5 \ 0.1]^T$, $\mathbf{m}_f = [0.6 \ 1.8 \ 0.7 \ 0.2 \ 0.2 \ 0.5 \ 0.3 \ 4.7 \ 0.4]^T$, $\boldsymbol{\sigma} = [0.4 \ 0 \ 0.2 \ 0.5 \ 0.3 \ 0.1 \ 0]^T$;
- s.2) $\Theta = 0.01$, $\mathbf{m}_0 = [1 \ 2 \ 0.001 \ 0.5 \ 0.1 \ 0.1 \ 0.1 \ 5 \ 0.1]^T$, $\mathbf{m}_f = [0.6 \ 1.8 \ 0.301 \ 0.2 \ 0.2 \ 0.5 \ 0.3 \ 4.7 \ 0.4]^T$, $\boldsymbol{\sigma} = [0.4 \ 0 \ 0.2 \ 0.5 \ 0.3 \ 0.1 \ 0]^T$;
- s.3) $\Theta = 0.1$, $\mathbf{m}_0 = [1 \ 2 \ 0.4 \ 0.5 \ 0.1 \ 0.1 \ 0.1 \ 5 \ 0.1]^T$, $\mathbf{m}_f = [0.6 \ 1.8 \ 0.7 \ 0.2 \ 0.2 \ 0.5 \ 0.3 \ 3 \ 2.1]^T$, $\boldsymbol{\sigma} = [2.1 \ 1.7 \ 1.9 \ 2.2 \ 2 \ 1.8 \ 0]^T$.
- s.4) $\Theta = 0.02$, $\mathbf{m}_0 = [1 \ 2 \ 1.4 \ 1.5 \ 1.1 \ 1.1 \ 1.1 \ 5 \ 1.1]^T$, $\mathbf{m}_f = [0.6 \ 1.8 \ 1.7 \ 1.2 \ 1.2 \ 1.5 \ 1.3 \ 3 \ 3.1]^T$, $\boldsymbol{\sigma} = [2.1 \ 1.7 \ 1.9 \ 2.2 \ 2 \ 1.8 \ 0]^T$.

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 \mathbf{R} and \mathbf{Q} , we suggest to use larger weights on matrix \mathbf{R} and smaller weights on matrix \mathbf{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 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 \mathbf{R} (i.e., putting more

4.4. Extended ON/OFF based methods

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 $\mathbf{Q} = q \cdot \mathbf{I}^{|P|}$, $\mathbf{R}[j, j] = r, \forall t_j \in T_p$.

	Control methods	Time steps	CPU time (ms)	Parameters
s.1)	appro. min-time	101	847	
	ON/OFF+	94	41	
	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	$N = 5, r = 1000, q = 1$
s.2)	appro. min-time	176	465	
	ON/OFF+	954	410	
	B-ON/OFF	132	192	$d = 2$ (or 1)
	MPC-ON/OFF	145	16,240	$N = 5, r = 1000, q = 1$
s.3)	appro. min-time	94	352	
	ON/OFF+	76	34	
	B-ON/OFF	76	115	$d = 20$ (or 15, 10)
	MPC-ON/OFF	75	52,803	$N = 10, r = 1000, q = 1000$
s.4)	appro. min-time	122	2,546	
	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 at 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 \mathbf{m}_f can be reached in h steps, now we verify if \mathbf{m}_f can also be reached in $k = h - 1$ steps, by solving LPP (4.16). If a control law is found, then we decrease k again and check if \mathbf{m}_f can still be reached; repeat this process until we find a k such that \mathbf{m}_f cannot be reached, then the minimum number of time steps to reach \mathbf{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: $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle, \mathbf{m}_f, \Theta$

Output: $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots$

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

$$\begin{aligned}
 \min \quad & Z = \|\mathbf{m}_k - \mathbf{m}_f\|_1 \\
 \text{s.t.} \quad & \mathbf{m}_{i+1} = \mathbf{m}_i + \Theta \cdot \mathbf{C} \cdot \mathbf{w}_i, \quad i = 0, 1, \dots, k-1 \\
 & \mathbf{w}_i[t_j] \leq \lambda_j \cdot \text{enab}(t_j, \mathbf{m}_i), \quad \forall t_j \in T, \quad i = 0, \dots, k-1 \\
 & \mathbf{w}_i \geq 0, \quad i = 0, \dots, k-1
 \end{aligned} \tag{4.16}$$

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

Remark 4.5.1. In problem 4.16, the minimization of $\|\mathbf{m}_k - \mathbf{m}_f\|_1$ can be turned into a LPP by introducing a new variable \mathbf{v} with constraints $\mathbf{v} \geq \mathbf{m}_k - \mathbf{m}_f, \mathbf{v} \geq -(\mathbf{m}_k - \mathbf{m}_f)$, then solving the problem by minimizing $\mathbf{1}^T \cdot \mathbf{v}$.

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 an 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

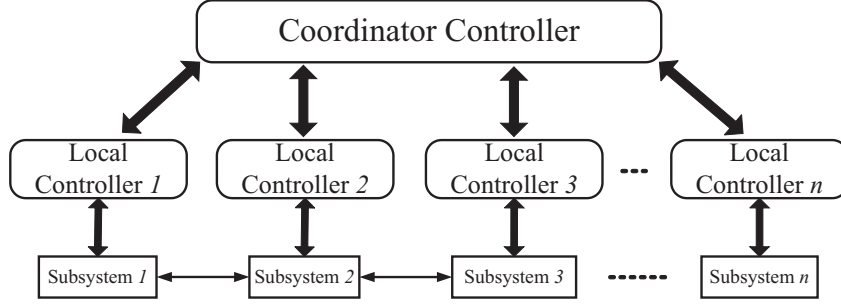


Figure 5.1: System Structures

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

5.3. Decentralized control of CF nets

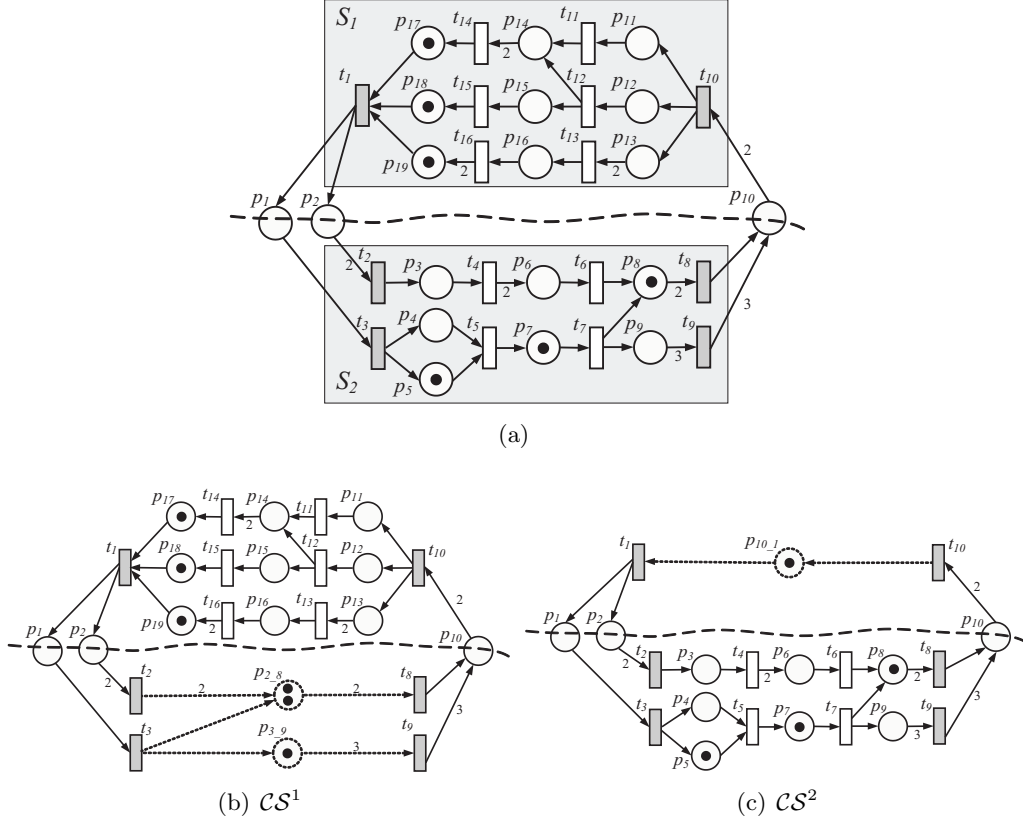


Figure 5.2: (a) A live and bounded CF net system and a cut $B = \{p_1, p_2, p_{10}\}$; (b) complemented subsystem CS^1 ; (c) complemented subsystem CS^2

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

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

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

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

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

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

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

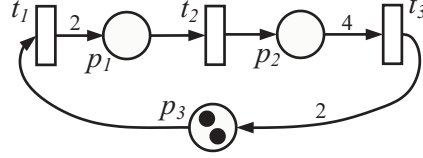


Figure 5.3: A simple CF net system with $\mathbf{m}_0 = [0 \ 0 \ 2]^T$

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

Assume that from \mathbf{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 $G(\pi) \cdot M(\pi, \mathbf{m})$.

Transition Reduction Rule (T-RR). Let t_j be a transition in a continuous CF net system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$, 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 $\mathcal{S}' = \langle \mathcal{N}', \mathbf{m}_0' \rangle$, 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 $\mathbf{m}_0'[p_{x-y}] = \mathbf{Post}(p_{x-y}, t_{in}) \cdot M(\pi, \mathbf{m}_0)$, 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 behaviors 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 $\mathbf{Post}(p_1, t_1) = 4$ and $\mathbf{Pre}(p_1, t_2) = 2$ (in this case, the marking of p_1 is still zero). Obviously, the overall behaviors of the system are not changed.

5.3. Decentralized control of CF nets

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} .

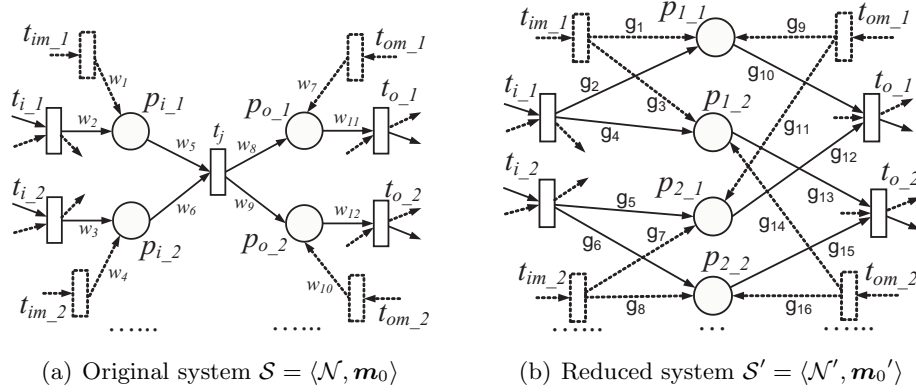


Figure 5.4: Transition reduction rule (T-RR): reducing t_j

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 $\mathbf{m}_0'[p_{1_1}] = g_2 \cdot M(\pi, \mathbf{m}_0)$.

Let $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$ and $\mathcal{S}' = \langle \mathcal{N}', \mathbf{m}_0' \rangle$ 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 \mathbf{m}_0 , the maximal enabling degree of $t_{o,1}$ by firing of σ_1 is:

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

In \mathcal{S}' , the enabling degree of $t_{o,1}$ under the initial marking is equal to:

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

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

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

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*

5.3. Decentralized control of CF nets

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 $\frac{m_0[p_1]}{Pre(p_1, t_{out})} \leq \frac{m_0[p_2]}{Pre(p_2, t_{out})}$, 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 $\frac{m_0[p_1]}{w_5} \leq \frac{m_0[p_2]}{w_6}$, then by removing p_2 , the reduced system is shown in Fig. 5.5(b).

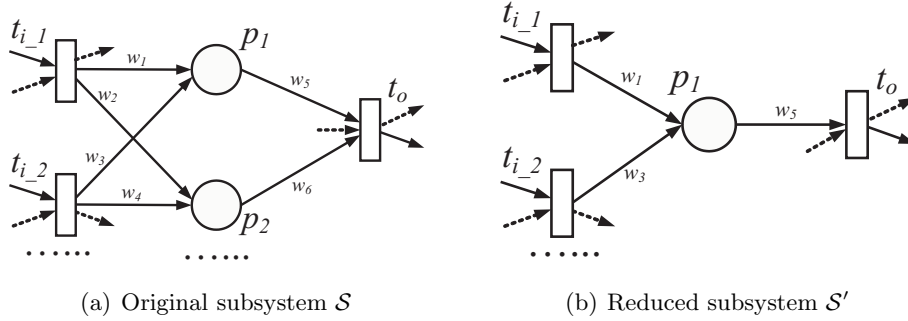


Figure 5.5: Place Reduction Rule (P-RR): reducing p_2

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 $\mathbf{m}_0'[p_{2,8}] = 2$, $\mathbf{m}_0'[p_{3,9}] = 1$. The reduced subsystem in Fig.5.6(e) is the abstraction of \mathcal{S}^2 .

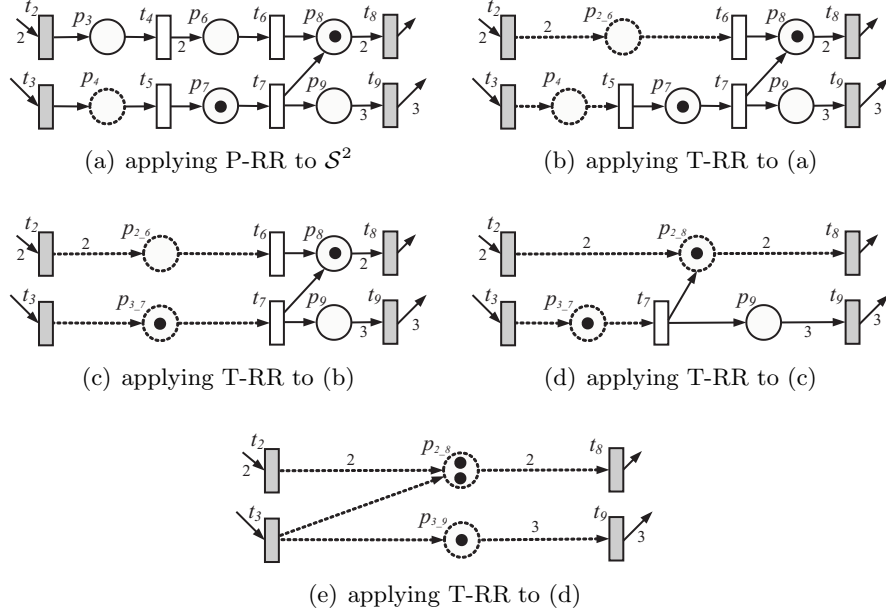


Figure 5.6: Reduction process of \mathcal{S}^2 in Fig. 5.2(a)

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.

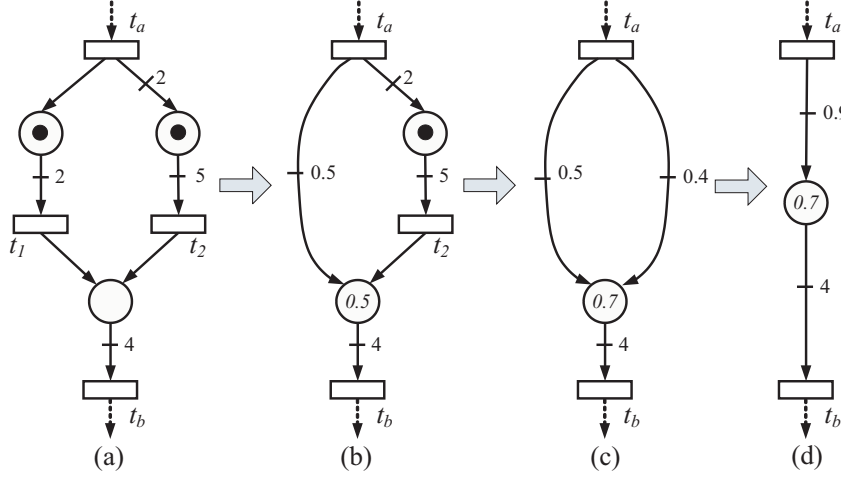


Figure 5.7: Reduction of the paths between t_a and t_b using T-RR: the classical reduction rules for discrete nets cannot be applied

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.

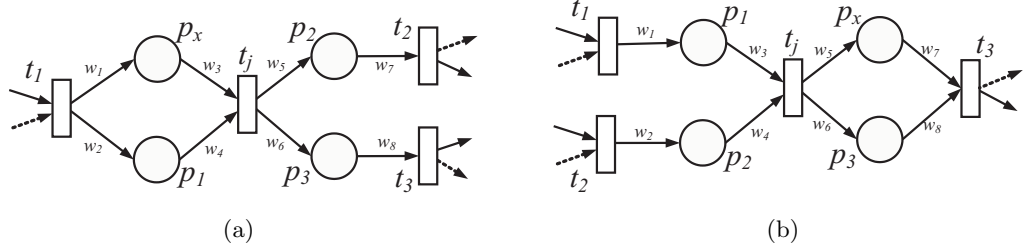
1) if A and B are both instances of T-RR (or P-RR), it is trivial.

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 $\mathbf{m}_0[p_x]/w_3 \geq \mathbf{m}_0[p_1]/w_4$. 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 $M(\pi_1, \mathbf{m}_0) \geq M(\pi_2, \mathbf{m}_0)$.

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 $\mathbf{m}_0'[p'_a] =$


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

$g_1 \cdot M(\pi_1, \mathbf{m}_0)$ and $\mathbf{m}_0'[p_a] = g_2 \cdot M(\pi_2, \mathbf{m}_0)$, therefore, $\frac{\mathbf{m}_0'[p'_a]}{g_5} \geq \frac{\mathbf{m}_0'[p_a]}{g_6}$, 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.

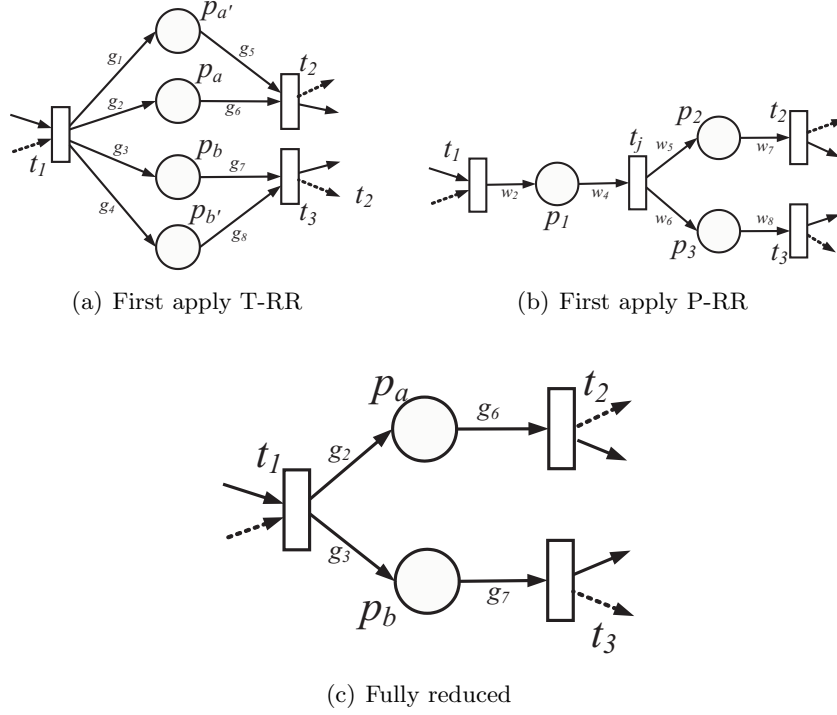


Figure 5.9: Reduction by applying rules in different order

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 \mathbf{m}_0 , \mathbf{m}_f of the original system, and its corresponding minimal firing count vector σ_{min} are shown in Table 5.1. As for the subsystems, the minimal firing count vectors σ_{min}^i of \mathcal{CS}^i for reaching the corresponding final marking \mathbf{m}_f^i from \mathbf{m}_0^i are computed separately, they are also given in Table 5.1. It can be observed that σ_{min}^1 and σ_{min}^2 are not compatible, because their interface transitions do not have the same firing counts, for instance, $\sigma_{min}^1[t_1] \neq \sigma_{min}^2[t_1]$.*

Let $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$ be the original system, with $\mathbf{m}_f > 0$ 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) σ_{min} : the minimal firing count vector driving \mathcal{S} to \mathbf{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) $\mathcal{CS}^i = \langle \mathcal{CN}^i, \mathbf{m}_0^i \rangle$: the complemented subsystems with corresponding final state \mathbf{m}_f^i , $i = 1, 2, \dots, K$.
- (5) \mathbf{x}^i : the minimal T-semiflow in \mathcal{CN}^i , $i = 1, 2, \dots, K$.
- (6) σ_{min}^i : the minimal firing count vector driving \mathcal{CS}^i to \mathbf{m}_f^i , $i = 1, 2, \dots, K$.

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., \mathbf{x}^i and σ_{min}^i are unique. So, any firing count vector σ^i driving \mathcal{CS}^i to its final state can be written as follows

$$\sigma^i = \sigma_{min}^i + \alpha^i \cdot \mathbf{x}^i, \quad \alpha^i \geq 0 \quad (5.1)$$