

Control of Continuous Petri Nets with

Infinite Servers Semantics

Hanife Apaydın Özkan

Ph.D. Dissertation

Graduate School of Sciences

Electrical and Electronics

Engineering Program

October 2010

ABSTRACT

Doctor of Philosophy Dissertation

CONTROL OF CONTINUOUS PETRI NETS WITH INFINITE SERVER SEMANTICS

HANİFE APAYDIN ÖZKAN

Anadolu University

Graduate School of Sciences

Electrical and Electronics Engineering Program

Supervisor: Associated Prof. Aydın Aybar

2010, 106 pages

In this thesis, timed continuous Petri net model is considered. It is necessary to investigate new control methods for timed continuous Petri nets because of their particular characteristics: synchronizations lead to switched systems of equations and input constraints that depend on the state. Even if their control problem may look easier than that of the original discrete net, in many cases the system is distributed and the controller cannot have access to all subsystems. In this case, a distributed controller should be considered. This thesis proposes three different aspects, all related to control of timed continuous Petri nets: 1) A heuristics for minimum time control which drives the system from the initial to target marking by minimizing the time through a linear or piecewise linear trajectory, 2) An online control method for the case that some components of target markings are not specified, 3) Distributed controller design for distributed model.

Keywords: Continuous Petri nets, control, heuristics, decentralized control.

ÖZET

Doktora Tezi

Dış Merkezli Sürekli Petri ağları için Kontrol Teknikleri

Hanife Apaydın Özkan

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Aydın Aybar

2010, 106 sayfa

Bu tezde, zamanlandırılmış sürekli Petri ağı modeli ele alınmıştır. Zamanlandırılmış sürekli Petri ağlarının kontrol girdilerinin durumlara bağlı olarak kısıtlanması gibi kendilerine özgü bazı özelliklerinden dolayı , bu modeller için yeni kontrol metodlarının geliştirilmesine ihtiyaç vardır. Zamanlandırılmış sürekli Petri ağlarının kontrolü, orjinal kesikli zaman Petri ağlarının kontrolüne göre daha kolay görünmekle beraber, ele alınan sistemin dağıtım (bağımlı) altsistemlerden oluşmuş olması durumunda, kontrolcü bütün altsistemlere erişemez. Bu durumda dağıtım kontrolcü yaklaşımının ele alınması gerekir. Bu tezde, zamanlandırılmış sürekli Petri ağlarının kontrolü üç farklı açıdan incelenmiştir: 1) Buluşsal bir kontrol yöntemi ile sistemin başlangıç durumundan hedef durumuna doğrusal ya da parçalı doğrusal bir yörünge üzerinde zamanı minimize ederek taşınması, 2) Hedef durumunun bazı bileşenlerinin belirtilmediği durumlar için çevrimiçi kontrolcü tasarımı, 3) Dağıtım modeller için dağıtım kontrolcü tasarımı.

Keywords: Sürekli Petri Ağları , kontrol, buluşsal yöntemler, dış merkezli kontrol.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Associated Prof. Dr. Aydın Aybar for the support of my Ph.D study and research. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Dr. Altuğ İftar, Associated Prof. Dr. Yusuf Oysal for their insightful comments, and hard questions.

The big part of this thesis was prepared during my stay in Zaragoza as a visiting researcher. My sincere thanks go to Prof. Dr. Manuel Silva for providing me the opportunity and proud of working in GISED and for his motivation, encouragement and immense knowledge. I am deeply grateful to my supervisors in Universidad de Zaragoza: Assistant Prof. Dr. Jorge Julvez and Assistant Prof. Dr. Cristian Mahulea. Both have effectively guided me with their great knowledge, positiveness, inspiration and enthusiasm. I would have been lost without them. Thanks to them also for their friendship, motivation and encouragement. I would also like to thank Assistant Prof. Dr. Nuray At, whom I consider to be my mentor. Her encouragement in the process of my application to Universidad de Zaragoza was invaluable.

I would like to thank all my friends in Spain and Turkey, especially, Dr. Renato Vazquez for helping me to understand the formalism of continuous Petri nets and answering my endless questions patiently, Ümmühan Başaran and Levid Wang for their friendship and understanding.

This work was partially supported by the European Communitys Seventh Framework Programme under project DISC (Grant Agreement n. INFISO-ICT-224498). Let me also thank Diputacin General de Aragón that provided me financial support during my 15 months stay in Zaragoza.

Last but not the least, I would like to thank my family: my father Nurullah Apaydın, my sister Hediye Özçelik, my niece Zübeyde Özçelik, my brother Emre Apaydın and my sweetheart husband Erkan Özkan for their unlimited love and support. I wish to dedicate this thesis to my precious family.



CONTENTS

ABSTRACT	i
ÖZET	ii
ACKNOWLEDGEMENT	iii
CONTENTS	iv
LIST OF FIGURES	vi
NOTATION	viii
1. INTRODUCTION	1
2. CONTINUOUS PETRI NETS	7
2.1 Untimed Discrete Petri nets	7
2.1.1 Structural properties	8
2.1.2 Behavioural properties	9
2.1.3 Petri net subclasses	9
2.2 Timed Discrete Petri nets	11
2.2.1 Stochastic Petri nets	11
2.3 Fluidification	12
2.4 Untimed Continuous Petri nets	12
2.5 Timed Continuous Petri nets	15
2.5.1 Servers semantics	15
2.6 Criteria for a good approximation	21
2.7 Summary	24
3. A HEURISTICS FOR MINIMUM TIME CONTROL OF TIMED CONTINUOUS PETRI NETS	26
3.1 Problem Statement	26
3.2 Controlled Timed Continuous Petri Nets	27



3.3	Computation of Linear Trajectories	27
3.3.1	$\mathbf{m}_0$ and $\mathbf{m}_f$ are in the same region	27
3.3.2	$\mathbf{m}_0$ and $\mathbf{m}_f$ are in different regions	28
3.4	Reducing the time by computing a piecewise linear trajectory .	33
3.4.1	Intermediate states on the borders	35
3.4.2	Interior intermediate states	35
3.5	Closed Loop Control	41
3.6	Case study	45
3.7	Summary	50

4.	A FINAL MARKING PLANNING METHOD FOR	
	JOIN FREE TIMED CONTINUOUS PETRI NETS	51
4.1	Join Free contPNs	51
4.2	Problem Statement	52
4.3	An Online Control Method for Join Free contPNs	52
4.4	Final Marking Planning for Join Free contPNs	57
4.5	Summary	58
5.	A CONTROL METHOD FOR DISTRIBUTED	
	TIMED CONTINUOUS PETRI NETS	60
5.1	Distributed Timed Continuous Petri nets	60
5.2	Problem Statement	62
5.3	A controller for DcontPN	64
5.3.1	Calculation of control inputs	72
5.4	Design of a Distributed Controller for General Structures	73
5.5	Case study	77
5.6	Summary	79
6.	CONCLUDING REMARKS	81
	REFERENCES	83
	APPENDIX	83

LIST OF FIGURES

2.1	Inclusions of subclasses [1]	10
2.2	A MTS net modeling an assembly line with Kanban strategy. .	11
2.3	(a)Continuous Petri net used in Example 2.1 (b)Reachability and lim-reachability sets	14
2.4	A timed continuous Petri net	17
2.5	A flexible manufacturing system taken from [2]	23
2.6	Throughput of transition E_M1_B for Example 2.5	24
3.1	A PN taken from [3]	30
3.2	Trajectory for Example 3.1	31
3.3	Evolution of markings for Example 3.1	34
3.4	Evolution of control inputs, controlled flows and and their upper bounds for Example 3.1	34
3.5	Trajectory for (a)1 intermediate state and (b)3 intermediate states for Example 3.2	39
3.6	Evolution of markings for Example 3.2	40
3.7	Evolution of control inputs for Example 3.2	40
3.8	Evolution of markings for Example 3.4	44
3.9	Evolution of control inputs for Example 3.4	44
3.10	Evolution of some markings without noise effect for the case study	47
3.11	Evolution of control inputs and controlled flows of some transi- tions without noise effect for the case study	48
3.12	Noise effect on transitions for the case study	48
3.13	Evolution of control inputs and controlled flows of some transi- tions with noise effect for the case study	49
3.14	Evolution of some markings with noise effect for the case study .	49
4.1	A JF Petri net taken from [4]	55
4.2	Evolution of markings for Example 4.1	56



4.3	Evolution of control action, $\mathbf{u}$, and controlled flow, $\mathbf{w}$, for Example 4.1	56
4.4	Evolution of markings for Example 4.2	58
4.5	Evolution of $\mathbf{u}$, $\mathbf{w}$ for Example 4.2	59
5.1	A DcontPN modeling a car manufacturing plant where p_a and p_b are channel places.	61
5.2	A simple DcontPN which is non-MTS	68
5.3	A DcontPN	69
5.4	A general structure for DcontPN system	75
5.5	A DcontPN used in Example 5.8.	77
5.6	A DcontPN modeling two manufacturing processes.	80

NOTATION

General

DES	Discrete Event System.
LPP	Linear Programming Problem.
BLP	Bilinear Programming Problem.
NLP	Nonlinear Programming Problem.
IPP	Integer Programming Problem.
RHC	Receding Horizon Control.
pdf	Probability density function.
$\mathbf{X}$	Matrix.
$\mathbf{x}$	Vector.
X_{ij}	Value in the i^{th} row and j^{th} column of matrix $\mathbf{X}$ ($X_{ij} = X(i, j)$).
x_i	Value of i^{th} entry of $\mathbf{x}$
$\mathbf{X}^T$	Transpose of matrix $\mathbf{X}$.
$rank(\mathbf{X})$	Rank of matrix $\mathbf{X}$.
$\mathbf{0}$	Vector $[0 \ 0 \dots 0]^T$.
$\mathbf{1}$	Vector $[1 \ 1 \dots 1]^T$.
$\mathbf{a} \geq \mathbf{b}$	For every i : $a(i) \geq b(i)$.
$\mathbf{a} > \mathbf{b}$	For every i : $a(i) > b(i)$.
$[a \ b]$	Interval from a to b including a and b .
$a \cdot b$	Algebraic product.
$\mathbb{N}$	Set of natural numbers, i.e., $\mathbb{N} = \{0, 1, 2, 3, \dots\}$.
$\mathbb{R}$	Set of real numbers.
$\mathbb{Z}$	Set of integers.
$\mathbb{Z}_{>0}$	Set of positive integers.
$\mathbb{R}_{\geq 0}$	Set of nonnegative real numbers.
$\mathbb{R}_{\geq 0}^n$	Nonnegative n dimensional euclidean space.
$t.u.$	Time unit.
$ \ast $	Size of the set $\ast$.

Petri Nets

PN	Discrete Petri net.
SPN	Stochastic Petri Net.
contPN	Timed continuous Petri net with infinite server semantics.
MTS	Mono T semiflow.
MG	Marked Graph
JF	Join Free
P	Set of places.
T	Set of transitions.
p_i	Place number i of the net.
t_j	Transition number j of the net.
Pre	$P \times T \rightarrow \mathbb{N}$ pre matrix.
Post	$P \times T \rightarrow \mathbb{N}$ post matrix.
C	Incidence matrix (C = Post - Pre).
$Pre(p_i, t_j)$	Value in the i^{th} row and j^{th} column of Pre ($Pre_{ij} = Pre(p_i, t_j)$).
$Post(p_i, t_j)$	Value in i^{th} row and j^{th} column of Post ($Post_{ij} = Post(p_i, t_j)$).
$C(p_i, t_j)$	Value in i^{th} row and j^{th} column of C ($C_{ij} = C(p_i, t_j)$).
$C(:, t_j)$	j^{th} column of C ($C_{:j} = C(:, t_j)$).
$C(p_i, :)$	i^{th} row of C ($C_{i:} = C(p_i, :)$).
$\mathbf{m}_0$	$P \rightarrow \mathbb{N}_{\geq 0}$ initial marking vector.
$\mathbf{m}(\tau)$	$P \rightarrow \mathbb{N}_{\geq 0}$ marking vector at time τ .
$m(\tau, p_i)$	Marking of place p_i at time τ ($m(\tau, p_i) = m_i$).
$\mathcal{N}$	Net structure consists of P , T , Pre , Post .
$R_D(\mathcal{N}, \mathbf{m}_0)$	Reachability set of PN $\langle \mathcal{N}, \mathbf{m}_0 \rangle$.
$\mathbf{x}$	$T \rightarrow \mathbb{N}$ T-semiflow vector.
$\mathbf{y}$	$P \rightarrow \mathbb{N}$ P-semiflow vector.
$\mathbf{B}_y$	Basis of P-semiflows.
$\mathbf{s}$	Flow integral vector.
s_f	Flow intagral value of transition $t_j \in T$.

Specific to continuous Petri nets

$f(\tau, t_j)$	Flow of transition t_j at time τ ($f(\tau, t_j) = f_j$).
λ	$ T \rightarrow \mathbb{R}_{\geq 0}^{ T }$ is firing rate or maximum firing speed vector.
$\mathbf{m}_0$	$P \rightarrow \mathbb{R}_{\geq 0}$ initial marking vector.
$\mathbf{m}(\tau)$	$P \rightarrow \mathbb{R}_{\geq 0}$ marking vector at time τ ($\mathbf{m} = \mathbf{m}(\tau)$).
$RS_C(\mathcal{N}, \mathbf{m}_0)$	Reachability set of continuous Petri net $\langle \mathcal{N}, \mathbf{m}_0 \rangle$.
$RS_{CL}(\mathcal{N}, \mathbf{m}_0)$	Lim-reachability set of continuous Petri net $\langle \mathcal{N}, \mathbf{m}_0 \rangle$.
$RS_{CL}^t(\mathcal{N}, \lambda, \mathbf{m}_0)$	Reachability set of timed continuous Petri net $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$.
$\mathcal{R}^z$	z^{th} region of the state space of a contPN.
$\eta(\mathbf{m})$	Label of the region that $\mathbf{m}$ belongs to.
$\mathcal{R}^{\eta(\mathbf{m})}$	Region that $\mathbf{m}$ belongs to.
Π^z	$T \times P \rightarrow \mathbb{R}_{\geq 0}$ Constraint matrix of region $\mathcal{R}^z$.
$\Pi^{\eta(\mathbf{m})}$	$T \times P \rightarrow \mathbb{R}_{\geq 0}$ Constraint matrix of the region $\eta(\mathbf{m})$.
$\mathcal{X}^I(t)$	Throughput of transition $t \in T$ of a contPN
$\mathcal{X}^F(t)$	Throughput of transition $t \in T$ of a timed continuous Petri net under finite server semantics

Specific to Distributed Continuous Petri Nets

DcontPN	Distributed timed continuous Petri net.
K	Set of subsystems.
P^k	Set of places of subsystem k .
T^k	Set of transitions of subsystem k .
$B^{(k,l)}$	Set of channel places directed from subsystem k to subsystem l .
$B^{(*,k)}$	Set of all input channels of subsystem k .
$B^{(k,*)}$	Set of all output channels of subsystem k .
$\mathbf{m}(\tau, P^k)$	Marking vector of subsystem k at time τ ($\mathbf{m}(P^k) = \mathbf{m}(\tau, P^k)$).

1. INTRODUCTION

A *Discrete Event System* (DES) is a dynamic system that evolves in accordance with the occurrence of physical events. DESs arise in the domains of traffic, manufacturing, robotics, telecommunication and communication systems [5, 6]. These kind of systems require control and coordination to ensure the orderly flow of events. Several formalisms and tools are developed for modeling and analysis of DESs, such as automata, Markov chains, Petri nets etc.

Discrete Petri Nets (PNs) are powerful graphical and mathematical tools for modeling, analysis and synthesis of DESs, that captures system properties as being concurrent, asynchronous, distributed, parallel non-deterministic and/or stochastic [7, 8]. The theoretic aspect of PNs provides a well defined mathematical theory for modeling and analysis of system behaviour while the graphical representation enables visualization of the state flow of a system. The power of PNs comes from the combination of these graphical and mathematical features.

The distributed state or marking of a PN is given by a vector of natural numbers which represent the number of tokens in each place. This is a significant advantage with respect to other formalisms such as automata, where the state space is a symbolic unstructured set.

Like in most modeling formalisms for DESs, PNs suffer from the so called *state explosion* which leads to an exponential growth of the size of state space with respect to the size of the system and population of initial state. For example, in production systems state explosion frequently occurs, when the markings associated to the number of parts in a buffer is very large. As a consequence, the analysis and control of these systems require a large amount of computational efforts, making enumerative analysis methods impractical.

Some relaxation techniques are studied to overcome the state explosion problem and to reduce the computational complexity of the analysis and synthesis of PNs (i.e. decomposition techniques, Lagrangian relaxations, fluidification and the others [9, 10, 11, 12]). *Fluidification* may be very useful

relaxation technique when applied to highly populated systems. The idea is to try to overcome, at least partially, the potentially very high computational complexity arising in many practical situations.

For PNs, fluidification was introduced in [9, 13] aiming at giving fluid (continuous) approximation of original PN in the sense of behaviours and properties, and these models are called *continuous Petri nets*.

In continuous Petri nets, in contrast to PNs, the firing of a transition can take in a real amount, hence a real number of tokens can be removed from the input places of the transition and a real number of tokens can be put in the output places. Therefore, the marking is not the vector of integers but a vector of real numbers. Hence, the set of reachable markings constitute a continuous region instead of a discrete set. Fluidification of PNs have some main advantages: 1) It allows to study PNs through a continuous approximated model (continuous Petri nets), thus avoiding the state explosion problem , 2) It provides higher computational efficiency, 3) For the analysis of continuous Petri nets, linear programming techniques were introduced instead of integer programming techniques in order to solve logical verification or performance evaluation problems [14]. Note that, *Linear Programming Problems* (LPPs) can be solved in polynomial time while *Integer Programming Problems* (IPPs) usually entail an exponential complexity. As a result, fluidification provides a significant computational gain.

In the real world, almost every event is time relevant, hence the necessity for including the time variable in the PN models is apparent [15, 16]. Furthermore, without an explicit notion of time it is not possible to analyze the performance (i.e., throughput and utilization) of the system. A realistic way to introduce time in PNs is to introduce delays according to some given probability density function (pdf). If the time delay is assumed to be a random variable with an exponential pdf, *Stochastic Petri Nets* (SPNs) are obtained [17, 18]. Due to the memoryless property of the exponential distribution and the race semantics, it was shown that reachability graph of a SPN is isomorphic to its associated continuous time Markov chain [19].

The concept of time can be introduced in continuous Petri nets in several ways leading timed continuous Petri nets. In the literature it is very common to associate time to transitions although some works associating time to places exist [20]. In this thesis, for the timing interpretation of continuous Petri nets, a first order approximation of SPNs, based on the mean value of the exponential pdf associated to the firing of the transitions is considered. Using these ideas, an approximated continuous and deterministic model is obtained from a discrete (stochastic) model. In continuous Petri nets, different ways of transition firing rules are derived leading firing semantics which defines the flow through transitions. Once the flow through transitions is known, the evolution of markings can be computed by using the fundamental state equation. In the literature, the most used two semantics are *finite servers semantics* [21, 22] and *infinite servers semantics* [9, 23].

Under finite servers semantics an event occurs when a place becomes empty and the flow through transitions is piecewise constant. Under infinite servers semantics an event is activated by a change in the place giving the minimum for the enabling degree of a transition and the flow through transitions is proportional to their enabling degree. These imply that timed continuous Petri nets with finite or infinite servers semantics are particular cases of *PieceWise Linear* (PWL) systems. In [24], it was shown that infinite servers semantics provides a better approximation of underlying discrete net than finite servers semantics for a wide class of Petri nets under some general conditions. Hence, timed continuous Petri nets with infinite servers semantics (contPN) are mainly focused in this thesis.

In control theory, the word *control* denotes to act, to put things in order to guarantee that the system behaves as desired. Since that is required most of the time, controller design is one of the most important research topics in system theory. Many works have been done related to control design for PWL systems [25, 26, 27]. But these control techniques can not be applied to contPN which has particular characteristics: synchronizations lead to switched systems of equations and input constraints that depend on the state. Consequently, it

is necessary to investigate new control methods for contPNs.

Different techniques have been proposed for control of contPNs in the literature [28, 4, 29, 30]. Steady state optimal control of contPN was studied in [31] where it is shown that, the optimal steady state control problem of contPN system can be solved by means of LPP in the case that all transitions are controllable and the objective function is linear. For the problem of reaching a given steady state from an initial marking, implicit and explicit *Model Predictive Control* (MPC) methods minimizing a certain performance index are proposed in [28]. The step tracking problem, i.e. design of control laws to drive the system states to target references was considered in [29] and a Lyapunov-function-based dynamic control algorithm was proposed for the problem. That method requires solving a *BiLinear Programming Problem* (BLP) for the computation of intermediate states.

For controller design, we follow the idea of slowing down the firing flow of transitions as in [28, 29, 31]. Hence, the control signals must be nonnegative (they can not force the speed of the system to exceed its nominal value) and upper bounded by functions of system states (applied control can at most stop the system).

In this thesis, we developed a control strategy for contPN which drives the system from initial marking to target marking by minimizing the time. In contrast to the method in [29], where step plus ramp trajectories under closed loop control are considered, the technique we propose for the computation of control law for direct linear trajectory is based on linear programming instead of nonlinear programming. Interestingly, although the complexity of this new approach is significantly lower, i.e., it can be solved in polynomial time, the time of the obtained trajectories are very similar in general. In order to obtain faster trajectories, intermediate states, not necessarily on the line connecting the initial and the target marking are computed by means of a BLP. These computations are then used to develop a closed loop scheme which takes discrete time representation [32] of the system.

In some systems, final states of some places are not as important as

minimizing the time to reach desired final states of some other places. Accordingly, corresponding components of the target state are not specified in the control problem of contPN model. For calculating the value of unspecified components under the objective of time minimization, the control strategy that we mentioned above requests solving *Nonlinear Programming Problem* (NLP). That is why we developed another control strategy. This strategy focuses on contPN without synchronizations called Join Free (JF) contPN as a starting step and drives JF contPN to a specified target state through a linear trajectory by means of LPP similar to the other method. But differently, this method proposes an online algorithm and requests solving BLP instead of NLP for calculating unspecified components.

The complexity of nowadays systems entails the consideration of more and more autonomous modes of operation for complex distributed plants. Each distributed plant is equipped with local controllers and local observers resulting in a networked embedded system that consists of subsystems and of a communication network connecting the various subsystems. We study the problem of building local controllers for large scale distributed discrete event systems. This problem has been studied before using automata [33, 34, 35] and PNs in [36, 37, 38, 39]. Even if the control problem on the continuous approximation may look easier than the one on the original discrete net, in many cases the system is distributed and the controller cannot have access to all subsystems. In this case, a distributed controller should be considered. We hence consider distributed timed continuous Petri nets (DcontPN). Such a system is composed by several timed continuous Petri net subsystems that model different parts of a plant, interconnected by buffers (or channels) modeled by places. These buffers contain the products produced by a given subsystem and needed by another subsystem. For the sake of simplicity we limit our study to the class of mono-T-semiflow net systems, but the results hold for a bigger class of systems with a very significant modeling power from a practical point of view, namely mono-T-semiflow reducible [40, 41] that is simply a generalization of mono-T-semiflow nets.

For the algorithms developed throughout this thesis, we developed MATLAB programs. These programs are implemented on a PC with Intel(R) Core(TM) 2CPU T5600 @ 1.83GHz, 2.00 GB of RAM. For some performance evolutions of continuous Petri nets and stochastic Petri nets, SimHPN (a MATLAB based simulator developed at Group of Discrete Event Systems Engineering Zaragoza, Spain) and TimeNET (a software package developed at the Real-Time Systems and Robotics group of Technische Universitt Berlin, Germany) are used, respectively.

The remainder of the thesis is organized as follows: In Chapter 2, some basic definitions and concepts related to discrete and continuous Petri nets are introduced. Infinite servers semantics and finite servers semantics are described. The reasonability of continuous approximation to discrete Petri nets is discussed in the sense of performance measure. In Chapter 3, the problem of controlling contPN is studied. The control methodology used on contPN is explained. A new control method that assigns piecewise constant flows to transitions in order to reach the target state through a linear or piecewise linear trajectory by minimizing the time is introduced. Then in order to minimize the noise effect, a closed loop control strategy that follows the trajectory computed by the open loop control is presented. In Chapter 4, an online control strategy for driving JF contPN to a desired target state is proposed. Then the case in which some components of target marking are not specified is dealt. In Chapter 5, first the control problem of driving a DcontPN consisting of two subsystems from an initial state to a target state is considered. Then the work extended to handle DcontPN structures composed by finite number of subsystems. Some conclusions and future directions are drawn in Chapter 6.

2. CONTINUOUS PETRI NETS

In this chapter, first the basic definitions and properties of untimed and timed discrete Petri nets are presented. Next, untimed and timed continuous Petri nets in which the number of marks in the places are real numbers instead of integer numbers are introduced and two different possibilities to introduce time in a continuous Petri net are described: Finite servers semantics and infinite servers semantics. Then, reasonable relaxation criteria for discrete Petri nets are addressed.

2.1 Untimed Discrete Petri nets

This section deals with autonomous discrete Petri nets and some useful notations and definitions.

Definition 2.1. A discrete Petri net (PN) system is a pair $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ where $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ is a net structure where $P = \{p_1, p_2, \dots, p_{|P|}\}$ and $T = \{t_1, t_2, \dots, t_{|T|}\}$ are the sets of places and transitions, respectively; $\mathbf{Pre}, \mathbf{Post} \in \mathbb{N}^{|P| \times |T|}$ are pre and post matrices connecting places and transitions; $\mathbf{m}_0 \in \mathbb{N}^{|P|}$ is initial marking vector (or initial state).

In the graphical representation of a PN, places are represented by circles and transitions are represented by bars. Places and transitions are connected by directed arcs. $Pre(p_i, t_j)$ or Pre_{ij} denotes the weight of arc directed from place $p_i \in P$ to transition $t_j \in T$ and $Post(p_i, t_j)$ or $Post_{ij}$ denotes the weight of arc directed from transition $t_j \in T$ to place $p_i \in P$. For a node (place or transition) $v \in P \cup T$, the sets of its input and output nodes are denoted by $\bullet v$ and $v^\bullet$, respectively.

$\mathbf{m} \in \mathbb{N}^{|P|}$ is a marking vector, $m(p_i)$ or m_i indicates the number of tokens in place $p_i \in P$. A transition $t_j \in T$ is enabled at $\mathbf{m}$ iff $m_i \geq Pre_{ij} \forall p_i \in \bullet t_j$ and its enabling degree $enab(t_j, \mathbf{m})$ is the value $\min_{p_i \in \bullet t_j} \left\{ \frac{m_i}{Pre_{ij}} \right\}$ rounded to nearest lower integer.

An enabled transition t_j can fire at marking $\mathbf{m}$ in a certain amount $\beta \in \mathbb{Z}_{>0}$ such that $\beta \leq \text{enab}(t_j, \mathbf{m})$.

If $\mathbf{m}$ is reachable from $\mathbf{m}_0$ through a finite sequence $\sigma = "t_{i1}, t_{i2}, t_{i3} \dots t_{ik}"$, the state (or fundamental) equation is satisfied:

$$\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \quad (2.1)$$

where $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ is the *token flow matrix* or *incidence matrix*, $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ is the *firing count vector*, i.e., σ_j is the cumulative amount of firings of t_j in the sequence σ .

Two types of properties can be studied with a PN model: *behavioural properties* which depends on initial marking and *structural properties* which does not depend on initial marking.

2.1.1 Structural properties

Annulers of the incidence matrix are important, because they provide an analytical characterization of various concepts of invariance underlying the net dynamics. Right and left natural annulers of the token flow matrix are called T- and P-semiflows, and denoted by $\mathbf{x}$ and $\mathbf{y}$, respectively. A semiflow $\mathbf{v}$ is said to be *minimal* when its support, $\|\mathbf{v}\|$, i.e., the set of its non-zero components, is not a proper superset of any other, and the greatest common divisor of its elements is one.

If $\mathbf{y} \geq 0$ is such that $\mathbf{y} \cdot \mathbf{C} = 0$ then, every marking $\mathbf{m}$ reachable from $\mathbf{m}_0$ satisfies:

$$\mathbf{y} \cdot \mathbf{m} = \mathbf{y} \cdot \mathbf{m}_0 + \mathbf{y} \cdot \mathbf{C} \cdot \boldsymbol{\sigma} = \mathbf{y} \cdot \mathbf{m}_0$$

This provides a *token balance law*. Analogously, if $\mathbf{x} \geq 0$ is such that $\mathbf{C} \cdot \mathbf{x} = 0$, then

$$\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{x} = \mathbf{m}_0$$

That is, T-semiflows correspond to potential repetitive sequences that bring the system from $\mathbf{m}_0$ back to $\mathbf{m}_0$. If $\mathbf{x} > 0$ exists such that $\mathbf{C} \cdot \mathbf{x} = 0$, the net is said to be *consistent*, and if $\mathbf{y} > 0$ exists such that $\mathbf{y} \cdot \mathbf{C} = 0$, the net is said to be *conservative*.

2.1.2 Behavioural properties

A marking $\mathbf{m}$ is said to be *reachable* from a marking $\mathbf{m}_0$ if there exists a sequence of firings that transforms $\mathbf{m}_0$ to $\mathbf{m}$ (see equation (2.1)). The set of all reachable markings from $\mathbf{m}_0$ is called *reachability set* and denoted by $R_D(\mathcal{N}, \mathbf{m}_0)$.

A place p_i is said to be *bounded* for an initial marking $\mathbf{m}_0$ if there is a natural integer k such that, for all markings reachable from $\mathbf{m}_0$, the number of tokens in p_i is not greater than k . A PN system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is *bounded* for an initial marking $\mathbf{m}_0$ if all places are bounded for $\mathbf{m}_0$.

A PN system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is *reversible* if, for each marking $\mathbf{m} \in R_D(\mathcal{N}, \mathbf{m}_0)$ $\mathbf{m}_0$ is reachable from $\mathbf{m}$.

A transition t_j is *live* for an initial marking $\mathbf{m}_0$, iff it can ultimately occur for every reachable marking, i.e. for every $\mathbf{m} \in R_D(\mathcal{N}, \mathbf{m}_0)$ there exists $\mathbf{m}' \in R_D(\mathcal{N}, \mathbf{m}_0)$ such that t_j is enabled in $\mathbf{m}'$. A PN system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is *live* if every transition is live.

A *deadlock* is a marking such that no transition is enabled. A PN system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is *deadlock-free* when any reachable marking enables some transition.

A *conflict* corresponds to the existence of a place which has at least two output transitions. More formally, in the case there exists $t_1, t_2 \in T$ such that $\bullet t_1 \cap \bullet t_2 \neq \emptyset$ it is said that t_1 and t_2 are in conflict (or structural conflict) relation.

2.1.3 Petri net subclasses

Petri net subclasses are defined by imposing some constraints on the structure of the net. Some of the most usual subclasses are given as follow:

- $\mathcal{N}$ is *ordinary* if all the arc weights are unitary, i.e., $\forall p \in P, \forall t \in T$, $Pre(p, t) \leq 1$ and $Post(p, t) \leq 1$;
- $\mathcal{N}$ is *Marked Graph* (MG) if it is ordinary and $\forall p \in P : |p^\bullet| = |\bullet p| = 1$ [42].

- $\mathcal{N}$ is a *Join Free* (JF) if $\forall t \in T, |\bullet t| = 1$.
- $\mathcal{N}$ is a *Choice Free* (CF) if $\forall p \in P, |p\bullet| = 1$ [43].
- $\mathcal{N}$ is a *Free Choice* (FC) net if it is ordinary and $\forall t, t' \in T$, if $\bullet t \cap \bullet t' \neq \emptyset$; then $\bullet t = \bullet t'$ [44].
- $\mathcal{N}$ is an *Equal Conflict* (EQ) net iff $\forall t_1, t_2 \in T$ such that $\bullet t_1 \cap \bullet t_2 \neq \emptyset$; $Pre(p, t_1) = Pre(p, t_2), \forall p \in P$.
- $\mathcal{N}$ is *Mono T-semiflow* (MTS) if it is conservative, consistent and has only one minimal T-semiflow [21].

The inclusion relation between MG, JF, CF, FC, EQ and MTS PNs has been represented in Figure 2.1.

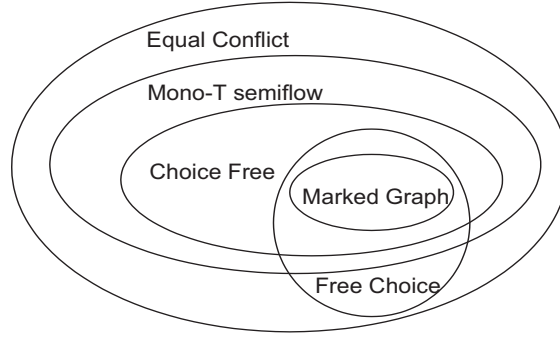


Figure 2.1: Inclusions of subclasses [1]

In this thesis, the work in Chapter 5 focuses on MTS class. The main properties of MTS nets are structural: Consistency, i.e., all transitions are covered by the unique minimal T-semiflow and conservativeness, i.e. all places are covered by P-semiflows. Hence, checking if a net belongs to the class has a polynomial time complexity. The modeling power of MTS is considerable since they represent a generalization of CF and MG PNs. In Figure 2.2 a MTS PN model is presented [45, 2]. This model represents an assembly line with Kanban¹ strategy.

¹A Kanban system is a means to achieve Just-in-time (JIT) production. It works on the basis that each process on a production line pulls just the number and type of components the process requires, at just the right time.

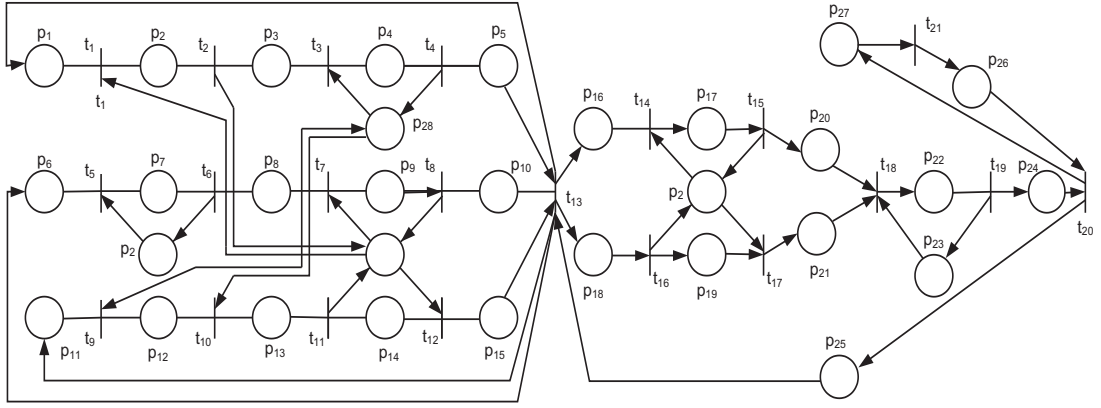


Figure 2.2: A MTS net modeling an assembly line with Kanban strategy.

2.2 Timed Discrete Petri nets

PNs were originally proposed without any notion of time or probability. But for many practical applications, the addition of time is a necessity. Since transitions represent activities and activities take time in most timed PN models, time is associated with transitions commonly. Such PN model is known as a *deterministic timed Petri net* if the delays are deterministically given, or as a *generalized stochastic net* if the delays are probabilistically specified.

Deterministic delays allow for simple analysis methods but have limited applicability. In real applications, delays correspond to the duration of activities which are typically variable. Therefore, fixed delays are often less appropriate. Hence most of the timed PN models use stochastic delays. In these models each delay is described by a probability distribution.

2.2.1 Stochastic Petri nets

For timed PN, to make analysis tractable typically only a restricted set of probability distributions is allowed. A way to introduce time in discrete PN is to assume that all transitions are timed with an exponential probability density function (pdf) which is a one parameter continuous distribution such that:

$$g(x) = \lambda \cdot e^{-\lambda \cdot x}, \quad x \geq 0 \quad (2.2)$$

Definition 2.2. A stochastic Petri net (SPN) system is a tuple $\langle \mathcal{N}, \boldsymbol{\lambda}, \mathbf{m}_0 \rangle$ where $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ is a net structure defined in Definition 2.1; $\boldsymbol{\lambda} \in \mathbb{R}_{\geq 0}^{|T|}$ is firing rate vector, such that $\lambda(t_j)$ or λ_j is the firing rate of transition t_j , $\mathbf{m}_0 \in \mathbb{N}^{|P|}$ is initial state.

If a transition models infinite servers, time to fire the transition t_j at a marking $\mathbf{m}$ is exponentially distributed random variable with parameter $\lambda_j \cdot \text{enab}(t_j, \mathbf{m})$ whose mean value is $\frac{1}{\lambda_j \cdot \text{enab}(t_j, \mathbf{m})}$.

If a transition models finite servers, time to fire the transition t_j at a marking $\mathbf{m}$ is exponentially distributed random variable with parameter λ_j whose mean value is $\frac{1}{\lambda_j}$.

2.3 Fluidification

PNs suffer from state explosion problem, like in all formalisms for DESs. One possible way to overcome this difficulty is fluidification (or continuization) as a classical relaxation technique. The resulting continuous Petri net systems may be studied by means of structural analysis [46]. Moreover, fluidification of PN allows the use of linear programming techniques (that can be solved in polynomial time) instead of integer programming techniques.

Unlike PN, a transition in a continuous Petri net can fire in a non-negative real quantity instead of in a natural quantity. Therefore, the transitions can be seen as valves through which fluid tokens flow, and the places can be seen as deposits in which this fluid is stored. Consequently, the continuous Petri net is a model in which the number of tokens in the places are real numbers instead of integers. Exhaustive enumeration techniques have no sense in continuous Petri nets, since the set of reachable markings is not a discrete set anymore, but a continuous region [47].

2.4 Untimed Continuous Petri nets

In this section, we will introduce the concept of autonomous continuous Petri nets.

Definition 2.3. A continuous Petri net system is a pair $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ where $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ is a net structure defined in Definition 2.1; $\mathbf{m}_0 \in \mathbb{R}^{|P|}$ is initial marking (state).

In continuous Petri nets, markings, $\mathbf{m}$, are not restricted to be integer, that is $\mathbf{m} \in \mathbb{R}_{\geq 0}^{|P|}$. $m(p_i)$ or m_i denotes the marking of place p_i . A transition $t_j \in T$ is enabled at $\mathbf{m}$ iff $\forall p_i \in \bullet t_j, m_i > 0$. That is, the enabling condition of continuous systems is the same as the enabling condition of discrete systems. As differ from the discrete case, the enabling degree is not limited to a natural number:

$$enab(t_j, \mathbf{m}) = \min_{p_i \in \bullet t_j} \left\{ \frac{m_i}{Pre_{ij}} \right\} \in \mathbb{R}_{>0}$$

An enabled transition t_j can fire in any real amount β , with $0 < \beta \leq enab(t_j, \mathbf{m})$ leading to a new state $\mathbf{m}' = \mathbf{m} + \beta \cdot \mathbf{C}(:, t_j)$ ($\mathbf{m} \xrightarrow{\beta t} \mathbf{m}'$) where $\mathbf{C}(:, t_j)$ is the column of transition t_j of token flow matrix $\mathbf{C}$.

The set of markings that are reachable with a finite firing sequence for a given system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ is denoted as $RS_C(\mathcal{N}, \mathbf{m}_0) = \{\mathbf{m} \mid \text{a finite fireable sequence } \sigma = “\beta_1 t_{a1} \dots \beta_k t_{ak}” \text{ exists such that } \mathbf{m}_0 \xrightarrow{\beta_1 t_{a1}} \mathbf{m}^1 \xrightarrow{\beta_2 t_{a2}} \mathbf{m}^2 \dots \xrightarrow{\beta_k t_{ak}} \mathbf{m}^k = \mathbf{m}\}$ where $t_{ai} \in T$ and $\beta_i \in \mathbb{R}_{>0}$ [48].

It is important to note that, $RS_C(\mathcal{N}, \mathbf{m}_0)$ is a convex region ². By assuming that infinitely long sequences can fire, reachability can be extended to *lim-reachability* for which the definition is given as follow:

Definition 2.4. [48] Let $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ be a continuous Petri net system. A marking $\mathbf{m} \in \mathbb{R}_{\geq 0}^{|P|}$ is *lim-reachable* iff a sequence of reachable markings $\mathbf{m}^i$, $i \geq 1$ exists such that

$$\mathbf{m}_0 \xrightarrow{\sigma_1} \mathbf{m}^1 \xrightarrow{\sigma_2} \mathbf{m}^2 \dots \mathbf{m}^{i-1} \xrightarrow{\sigma_i} \mathbf{m}^i \dots$$

and $\lim_{i \rightarrow \infty} \mathbf{m}^i = \mathbf{m}$. The *lim-reachable* set is denoted by $RS_{CL}(\mathcal{N}, \mathbf{m}_0)$.

²Let $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ be a continuous system and $\mathbf{m}^1$ and $\mathbf{m}^2$ two reachable markings, i.e. $\mathbf{m}_0 \xrightarrow{\sigma_1} \mathbf{m}^1$ and $\mathbf{m}_0 \xrightarrow{\sigma_2} \mathbf{m}^2$. Then any marking $\mathbf{m}^3 = \alpha \cdot \mathbf{m}^1 + (1 - \alpha) \cdot \mathbf{m}^2$, $\forall \alpha \in [0, 1]$ is also reachable from $\mathbf{m}_0$.

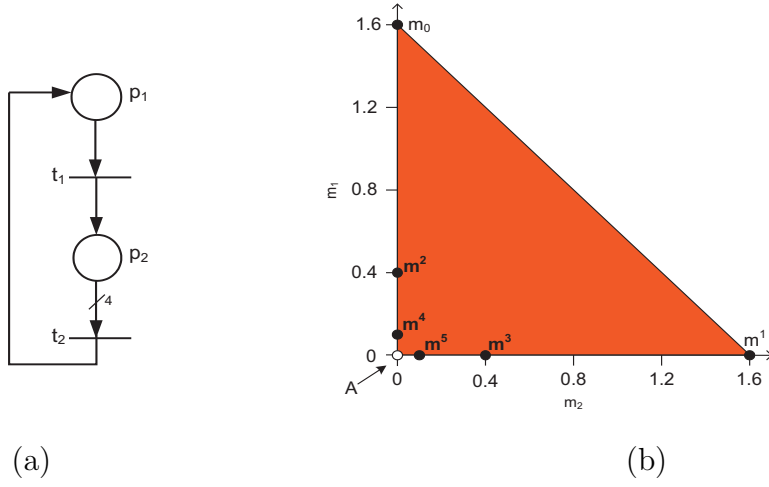


Figure 2.3: (a) Continuous Petri net used in Example 2.1 (b) Reachability and lim-reachability sets

Example 2.1. Let us consider the continuous Petri net in Figure 2.3 with $\mathbf{m}_0 = [1.6 \ 0]^T$. At this marking t_1 can fire. The firing of t_1 in the amount 1.6 leads the new marking $\mathbf{m}^1 = [0 \ 1.6]^T$ from which t_2 can fire in amount of 0.4 leading to $\mathbf{m}^2 = [0.4 \ 0]^T$. Now firing t_1 in an amount of 0.4 followed by t_2 ($\mathbf{m}^3 = [0 \ 0.4]^T$) in the maximum amount 0.1 the system evolves to $\mathbf{m}^4 = [0.1 \ 0]^T$. Let us continue to fire the same pair of transitions t_1 and t_2 , the marking obtained at step k be denoted by $\mathbf{m}^k$. Then it is obvious that $\lim_{k \rightarrow \infty} \mathbf{m}^k = [0 \ 0]^T = \mathbf{m}$. Therefore, the marking $\mathbf{m}$ is reached in the continuous Petri net system at the limit. In other words, the net system reaches deadlock in the limit.

In Figure 2.3(b), the reachability set and lim-reachability set for continuous Petri net in Figure 2.3(a) are shown. The only difference between these two sets are is point A that belongs to the $RS_{CL}(\mathcal{N}, \mathbf{m}_0)$ but not $RS_C(\mathcal{N}, \mathbf{m}_0)$.

For consistent nets, lim-reachability condition can be relaxed to a linear inequality system.

Proposition 2.1. [49, 48] Let $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ be a continuous Petri net system. If it is consistent and all transitions are fireable the following statements are equivalent:

- $\mathbf{m}$ is lim-reachable, i.e. $\mathbf{m} \in RS_{CL}(\mathcal{N}, \mathbf{m}_0)$.

- $\exists \mathbf{x} \geq 0$ such that $\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{x}$
- $\mathbf{B}_y^T \cdot \mathbf{m} = \mathbf{B}_y^T \cdot \mathbf{m}_0, \mathbf{m} \geq 0$ where $\mathbf{B}_y$ is a basis of P -semiflows.

2.5 Timed Continuous Petri nets

In order to include timing interpretation to continuous Petri nets we use a first order (or deterministic) approximation of SPNs, assuming that the delays associated to the firing of transitions can be approximated by their mean values. This leads continuous and deterministic “approximated” model [50].

Definition 2.5. A (deterministically) timed continuous Petri net system $\langle \mathcal{N}, \boldsymbol{\lambda}, \mathbf{m}_0 \rangle$ is a continuous Petri net system $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ together with a vector $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^{|T|}$ where $\lambda(t_j)$ or λ_j is the firing rate or maximum firing speed (depending on the semantics) of transition t_j .

As in PNs, the state equation (2.1) summarizes the way the marking evolves along time. The state equation of continuous Petri net has an explicit dependence on time $\mathbf{m}(\tau) = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}(\tau)$ where τ is global time. But, in continuous systems, the marking is continuously changing, so we may consider the derivative of $\mathbf{m}$ with respect to time. This way, $\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$ is obtained. Here, $\dot{\boldsymbol{\sigma}}(\tau)$ is flow through transitions and it is denoted by $\mathbf{f}(\tau) = \dot{\boldsymbol{\sigma}}(\tau)$. Hence, the state equation is

$$\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \mathbf{f}(\tau) \quad (2.3)$$

For the sake of simplicity τ is omitted in the rest of the thesis.

2.5.1 Servers semantics

In SPNs, for the definition of flow $\mathbf{f}$ different servers semantics have been defined, the two most important being finite servers (or constant speed) and infinite servers (or variable speed) [9, 21]. In this section we will consider deterministic first-moment approximations of these semantics for continuous systems.

- Infinite servers semantics: In SPN systems, infinite-servers semantics means that the delay between the enabling and the firing of a transition t_j at a marking $\mathbf{m}$ is the minimum of $enab(t_j, \mathbf{m})$ number of independent exponentially distributed variables, all of them with the same parameter λ_j . This yields an exponential distribution with parameter $\lambda_j \cdot enab(t_j, \mathbf{m})$. By making first-moment deterministic approximation, i.e., we just take into account the mean values, the firing of the transition takes $\frac{1}{\lambda_j \cdot enab(t_j, \mathbf{m})}$ time units. λ_j is called as *firing rate* of transition t_j . The flow of transition t_j , $f(t_j)$ or f_j is defined as:

$$f_j = \lambda_j \cdot enab(t_j, \mathbf{m}) = \lambda_j \cdot \min_{p_i \in \bullet t_j} \left\{ \frac{m_i}{Pre_{ij}} \right\} \quad (2.4)$$

- Finite servers semantics: In SPN systems, finite-servers semantics means that the delay between the enabling and the firing of a transition t_j at a marking $\mathbf{m}$ is exponentially distributed variable, with parameter λ_j . By making first-moment deterministic approximation, i.e., we just take into account the mean values, the firing of the transition takes $\frac{1}{\lambda_j}$ time units. For timed continuous Petri net, λ_j is called as *maximal firing speed* of transition t_j . If the markings of the input places of a transition are strictly greater than zero, its flow will be λ_j . Otherwise, the flow will be the minimum between its maximal firing speed and the total input flow of the places with zero marking. Hence, the flow of transition t_j , $f(t_j)$ or f_j is defined as:

$$f_j = \begin{cases} \lambda_j & , \text{ if } \nexists p_i \in \bullet t_j \text{ with } m_i = 0 \\ \min \left\{ \min_{p_i \in \hat{P}} \left\{ \sum_{t_k \in \bullet p_i} \frac{f_k \cdot Post_{ik}}{Pre_{ij}} \right\}, \lambda_j \right\} & , \text{ otherwise} \end{cases} \quad (2.5)$$

where $\hat{P} = \{p_i \mid p_i \in \bullet t_j \text{ and } m_i = 0\}$

In timed continuous Petri nets a transition is like a station in Queuing Networks: the meeting point of servers and clients. If a transition can be fluidified, the more appropriate firing semantics depends on the number of servers and clients. Assuming that there may be many or few of each one

Table 2.1: The four cases for possible continuization of a transition [23]

Clients	Servers	Semantics of the transition
few(D)	few	Discrete transition
many(C)	few	Finite servers semantics (bounds to firing speed)
few(D)	many	Discrete transition (servers become implicit places)
many(C)	many	Infinite servers semantics (speed is enabling-driven)

of them, continuization can be considered for clients, for servers or for both. Table 2.1 represents the four theoretically possible cases. Suggested semantics of transitions are also given according to the population of servers and clients.

Example 2.2. *Let us consider continuous relaxed form of PN in Fig.2.4 with $\mathbf{m} = [2 \ 0 \ 0 \ 1]^T$. The flow of transitions will be:*

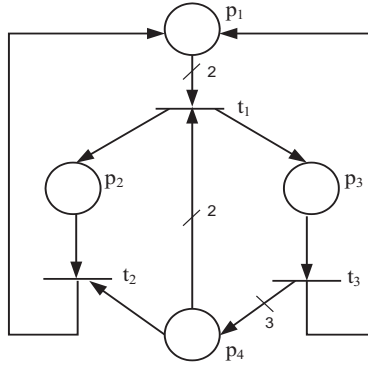


Figure 2.4: A timed continuous Petri net

- *Under infinite servers semantics*

$$\begin{aligned}
 f_1 &= \lambda_1 \cdot \min\left\{\frac{m_1}{2}, \frac{m_4}{2}\right\} \\
 f_2 &= \lambda_2 \cdot \min\{m_2, m_4\} \\
 f_3 &= \lambda_3 \cdot m_3
 \end{aligned} \tag{2.6}$$

- *Under finite servers semantics*

$$\begin{aligned}
 m_1 > 0 &\Rightarrow f_1 = \lambda_1 \\
 m_2 = 0 &\Rightarrow f_2 = \min\{\lambda_2, f_1\} = \min\{\lambda_2, \lambda_1\} \\
 m_3 = 0 &\Rightarrow f_3 = \min\{\lambda_3, f_1\} = \min\{\lambda_3, \lambda_1\}
 \end{aligned} \tag{2.7}$$

Both infinite servers semantics and finite servers semantics have piece-wise linear behaviour. Under finite servers semantics the behaviour is changed when a place is emptied, under infinite servers semantics the change happens when the place gives the minimum marking into synchronization is changed. In all the cases switching between linear system is given by internal event.

In order to decide the best semantics to use in given case some work have been done, i.e. [21, 24]. It has been proved that the continuous model under infinite servers semantics provides a better approximation of the original discrete model under some general conditions. Therefore, infinite servers semantics is mainly focused in this thesis.

In a timed continuous Petri net under infinite servers semantics (contPN), the flow of a transition depends on its enabling degree which is based on the minimum function (see equation (2.4)). The min in the definition leads to the concept of *configurations*:

Definition 2.6. *A configuration of a contPN is a set of (p, t) arcs, one per transition covering the set of transitions T . It assigns to each transition one place which is constraining that transition.*

The maximum number of configurations is calculated by

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

Example 2.3. *For the system sketched in Figure 2.4, the flow of t_1 can be restricted by the marking of p_1 or p_4 and the flow of t_2 can be restricted by the marking of p_2 or p_4 . Hence the system has maximum 4 number of configurations:*

- $\mathcal{C}^1 = \{(p_1, t_1), (p_2, t_2), (p_3, t_3)\}$
- $\mathcal{C}^2 = \{(p_4, t_1), (p_2, t_2), (p_3, t_3)\}$
- $\mathcal{C}^3 = \{(p_1, t_1), (p_4, t_2), (p_3, t_3)\}$
- $\mathcal{C}^4 = \{(p_4, t_1), (p_4, t_2), (p_3, t_3)\}$



The reachability set of a contPN $(RS_{CL}^t(\mathcal{N}, \lambda, \mathbf{m}_0))$ can be partitioned, associating to each configuration $\mathcal{C}^k$ a region $\mathcal{R}^k$ such that $RS_{CL}^t(\mathcal{N}, \mathbf{m}_0) = \mathcal{R}^1 \cup \dots \cup \mathcal{R}^\gamma$. Intuitively, each $\mathcal{R}^z$ denotes a *region* where the flow is limited by the same subset of places (one for each transition), that is a region $\mathcal{R}^z$ is defined as the set of markings such that if p_i is constraining place of transition t_j $((p_i, t_j) \in \mathcal{C}^z)$ then,

$$\forall \mathbf{m} \in \mathcal{R}^z, \quad \frac{m_i}{Pre_{ij}} = \min_{p_k \in \bullet t_j} \left\{ \frac{m_k}{Pre_{kj}} \right\}$$

These regions only may have borders in common, but their insides are disjoint.

Using the notions of *configuration* and *region*, a linear system is associated to each region, that linear system will govern the evolution of the contPN when the marking belongs to that region. This can be done since in each region, the same set of places gives the enabling degree of the transitions.

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

$$\Pi_{ji}^z = \begin{cases} \frac{1}{Pre_{ij}}, & \text{if } p_i \text{ is constraining } t_j \\ 0, & \text{otherwise} \end{cases} \quad (2.8)$$

In this thesis, $\eta(\mathbf{m})$ denotes the function which gives the label of the region (or configuration) that $\mathbf{m}$ belongs to. If marking $\mathbf{m}$ belongs to $\mathcal{R}^z$, we can denote $\mathbf{\Pi}^{\eta(\mathbf{m})} = \mathbf{\Pi}^z$ the corresponding constraint matrix (if $\mathbf{m}$ is a border point and belongs to several regions, $\mathbf{\Pi}^{\eta(\mathbf{m})}$ is the constraint matrix of any of these regions). In the constraint matrix, each row $j = 1, 2, \dots, |T|$ has only one non-null element in the position i that corresponds to the place p_i that restricts the flow of transition t_j . Furthermore, the firing rate of transitions can also be represented by a diagonal matrix $\mathbf{\Lambda} : T \times T \rightarrow \mathbb{R}_{>0}^{|T|}$

$$\Lambda_{jh} = \begin{cases} \lambda_j, & \text{if } j = h \\ 0, & \text{otherwise} \end{cases} \quad (2.9)$$

Using this notation, the nonlinear flow of the transitions at a given state $\mathbf{m}$ can be written as:

$$\mathbf{f} = \mathbf{\Lambda} \cdot \mathbf{\Pi}^{\eta(\mathbf{m})} \cdot \mathbf{m} \quad (2.10)$$

Finally, *state equation* of uncontrolled contPN is as follow:

$$\dot{\mathbf{m}} = \mathbf{C} \cdot \mathbf{f} = \mathbf{C} \cdot \mathbf{\Lambda} \cdot \mathbf{\Pi}^{\eta(\mathbf{m})} \cdot \mathbf{m} \quad (2.11)$$

Example 2.4. As we showed in Example 2.3, the maximum 4 number of regions exist ($\gamma = 4$) in the state space of the system in Figure 2.4. For $m_1 \leq m_4$ and $m_2 \leq m_4$, the active configuration is $\mathcal{C}^1 = \{(p_1, t_1), (p_2, t_2), (p_3, t_3)\}$.

If a marking $\mathbf{m}$ satisfies $m_1 \leq m_4$ and $m_2 \leq m_4$ (i.e. $\eta(\mathbf{m}) = 1$), then $\mathbf{m} \in \mathcal{R}^1$ with the constraining matrix $\mathbf{\Pi}^1$, i.e. $\Pi_{11}^1 = \frac{1}{Pre_{11}} = 0.5$, $\Pi_{21}^1 = \frac{1}{Pre_{12}} = 1$, $\Pi_{33}^1 = \frac{1}{Pre_{33}} = 1$, $\Pi_{41}^1 = \frac{1}{Pre_{14}} = 0.5$ while all other entities of $\mathbf{\Pi}^1$ are zero.

Constraint matrix of all 4 regions are calculated by using (2.8) and they are given as follow:

$$\begin{aligned} \bullet \mathcal{R}^1: m_1 \leq m_4 \text{ and } m_2 \leq m_4 \text{ with } \mathbf{\Pi}^1 &= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5 & 0 & 0 \end{bmatrix} \\ \bullet \mathcal{R}^2: m_4 \leq m_1 \text{ and } m_2 \leq m_4 \text{ with } \mathbf{\Pi}^2 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5 & 0 & 0 \end{bmatrix} \\ \bullet \mathcal{R}^3: m_1 \leq m_4 \text{ and } m_4 \leq m_2 \text{ with } \mathbf{\Pi}^3 &= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ \bullet \mathcal{R}^4: m_4 \leq m_1 \text{ and } m_4 \leq m_2 \text{ with } \mathbf{\Pi}^4 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0.5 & 1 & 0 \end{bmatrix} \end{aligned}$$



Let us assume that $\lambda = [3 \ 1 \ 1]^T$. Then the system is ruled by the following set of systems of the form:

$$\dot{m} = \begin{cases} C \cdot \Lambda \cdot \Pi^1 \cdot m = \begin{bmatrix} -3 & 1 & 1 & 0 \\ 1.5 & -1 & 0 & 0 \\ 1.5 & 0 & -1 & 0 \\ -3 & -1 & 3 & 0 \end{bmatrix} \cdot m, & \text{if } m \in \mathcal{R}^1 \\ \\ C \cdot \Lambda \cdot \Pi^2 \cdot m = \begin{bmatrix} -3 & 0 & 1 & 1 \\ 1.5 & 0 & 0 & -1 \\ 1.5 & 0 & -1 & 0 \\ -3 & 0 & 3 & -1 \end{bmatrix} \cdot m, & \text{if } m \in \mathcal{R}^2 \\ \\ C \cdot \Lambda \cdot \Pi^3 \cdot m = \begin{bmatrix} 0 & 1 & 1 & -3 \\ 0 & -1 & 0 & 1.5 \\ 0 & 0 & -1 & 1.5 \\ 0 & -1 & 3 & -3 \end{bmatrix} \cdot m, & \text{if } m \in \mathcal{R}^3 \\ \\ C \cdot \Lambda \cdot \Pi^4 \cdot m = \begin{bmatrix} 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & -1 & 1.5 \\ 0 & 0 & 3 & -4 \end{bmatrix} \cdot m, & \text{if } m \in \mathcal{R}^4 \end{cases} \quad (2.12)$$

2.6 Criteria for a good approximation

Continuous Petri nets are used to obtain an approximation for behaviour of discrete Petri nets [51, 52]. Hence, one may suppose that the basic qualitative properties (deadlock-freeness, liveness,.. etc.) are preserved. But this is not always true [23, 48]. For example, liveness of the continuous Petri net is neither necessary nor sufficient for liveness of the underlying discrete Petri net [53, 14]. This fact puts a limit to this relaxation approach.

The reasonable continuization of Petri nets requires that qualitative properties are preserved if the initial marking is multiplied by a natural number k .

Which means if a PN with $\mathbf{m}_0$ satisfies a certain property then the same net with $k \cdot \mathbf{m}_0$ should satisfy, too.

A performance measure that is often used in discrete Petri net systems is the *throughput* of a transition in the steady state, i.e., the number of firings per time unit. In the view of continuous Petri nets, this corresponds to the flow of the transition. Throughput is an important performance index of the production systems, since it is possible to determine important information by this index, i.e. liveness.

The throughput of a continuous Petri net is not identical to the throughput of the underlying discrete Petri net. It is often but not always an upper bound of this throughput [54]. This is because throughput is non-monotonic w.r.t. initial marking scaling. In the view of throughputs, two main properties must be satisfied for monotonicity w.r.t. initial marking scaling [9]:

- Let $\mathcal{X}^k(t)$ be the the throughput of transition t in steady state, when the initial marking is $\mathbf{m}^k(0) = k \cdot \mathbf{m}_0$. Then $(\mathcal{X}^k(t)/k) \geq \mathcal{X}^1(t)$.
- The previous sequence is upper bounded by throughput of the continuous Petri net system, i.e. $(\mathcal{X}^k(t)/k) \leq \mathcal{X}^I(t)$ or $\mathcal{X}^k(t) \leq \mathcal{X}^F(t)$

In Example 2.5, behaviour of a timed discrete Petri net which models a manufacturing system and its continuous approximation are compared by means of throughput.

Example 2.5. *The SPN in Figure 2.5 represents a flexible manufacturing system (taken from [2]) with $\lambda = 1$. The system produces final product by composing two different parts, A and B. Parts A are processed in machine M1 and then in machine M2 while parts B are processed in machine M2 and after in machine M1. The intermediate products are stored in B_1A and B_1B, and the final products respectively in B_2A and B_2B. Machine M3 assembles part A and part B. By this way, the final product which is temporally stored in B_3 is produced. The parts are moved on pallets. Hence, in this system we have three sequential machines, 4 intermediate deposits and one terminal*

deposit. At initial time, initial marking is $m_0(Pallets_A) = m_0(Pallets_B) = 10$; $m_0(M1_Idle) = m_0(M2_Idle) = m_0(M3_Idle) = m_0(B_3_Empty) = 1$; $m_0(Max_A) = m_0(Max_B) = 5$ and the rest unmarked.

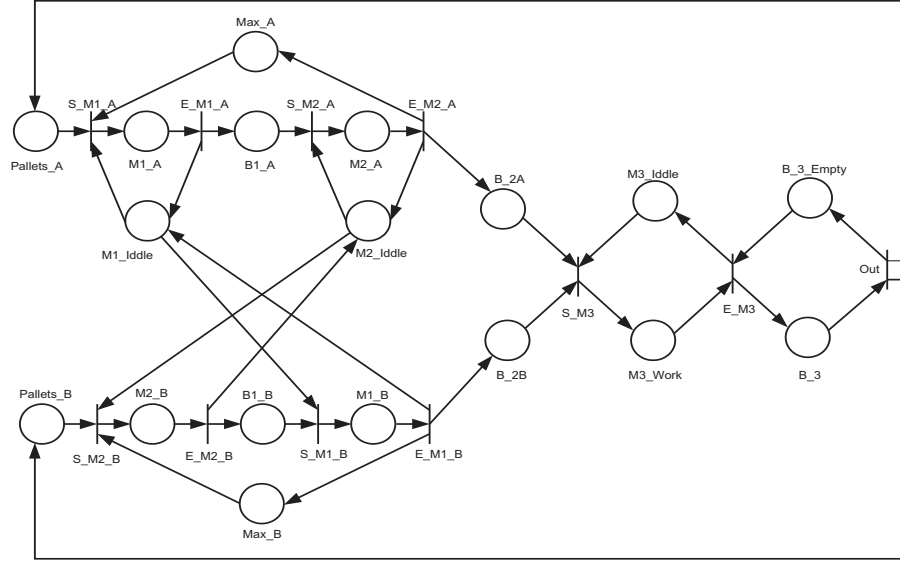


Figure 2.5: A flexible manufacturing system taken from [2]

The net has been analyzed using infinite servers semantics and finite servers semantics (single-server). Table 2.2 shows for the SPN system how the size of reachability set increases and the value that is obtained for the throughput changes when the initial marking is scaled. Clearly, the size of the reachability set is growing by increasing the marking of initial state. Analysis techniques based on the enumeration is difficult to apply with this huge reachability set.

According to the throughput results, under infinite servers semantics, for every k , $(\mathcal{X}^k(E_M1_B)/k) \leq (\mathcal{X}^{k+1}(E_M1_B)/(k+1))$ and moreover $\lim_{k \rightarrow \infty} (\mathcal{X}^k(E_M1_B)/k) \approx \mathcal{X}^I(E_M1_B)$, i.e. the throughput per copy of $\mathbf{m}_0$ is monotonically increasing and tends to $\mathcal{X}^I(E_M1_B)$. In other words, as the marking of the net is multiplied by a constant k , the throughput (normalized in the infinite servers case) approaches to the one obtained when the net is seen as continuous.

Obviously, for this net contPN with infinite servers semantics provides

a reasonable approximation to original discrete net.

Table 2.2: Throughput of E_M1_B when the initial marking is $k \cdot \mathbf{m}_0$

k	Size of reachability set	Infinite servers		Finite servers
		$\mathcal{X}_I$	$\mathcal{X}_I/k$	$\mathcal{X}_F$
1	537	0.3168	0.3168	0.316
2	2135	0.6466	0.3233	0.323
3	10850	0.9817	0.3250	0.333
4	379456	1.6422	0.3290	0.333
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Continuous	∞	∞	0.3308	1

In the graphics showed in Figure 2.6, the throughput course of the transition E_M1_B respect to the number of tokens in the initial marking of $Pallets_A$ and $Pallets_B$ reported. Throughputs of infinite servers semantics approximates very well to that of SPN.

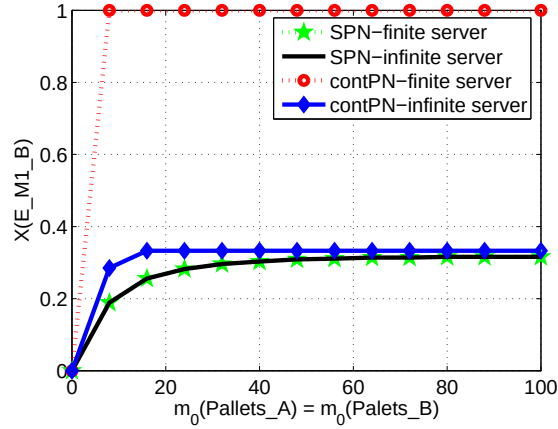


Figure 2.6: Throughput of transition E_M1_B for Example 2.5

2.7 Summary

Continuization may be a very useful relaxation technique when applied to systems with high traffic, independently of its nature or application domain.

This relaxation avoids the state explosion problem which is a fundamental problem in the analysis and sythesis of DESs. In continuos Petri nets, as differ from PNs, the amount in which a transition is fired is real numbers instead of natural numbers. A first order approximation of the discrete Petri net is used to define the flow through transitions which defines the evolution of the system.

Continuous Petri nets have potential for the application of more analytical techniques, possibly at the price of losing some modeling and analysis capabilities with respect to discrete view. Although it seems reasonable to expect that the continuized model and the original discrete models have a close behavior, it is not always true. For example, by means of quantitative properties some precision may be lost, and even qualitative properties of the discrete system may not be preserved by the continuous one. Furthermore, since continuization implies removing a constraint (the integrality of the firing), it could be thought that in timed systems the throughput of the continuous system is an upper bound of the discrete [54]. However, this is not always the case [23, 48]. However, some properties are preserved. For example, a live and bounded continuous system is structurally live and structurally bounded as discrete [9], for a marking to be reachable in the discrete model, reachability in the continuous one must be guaranteed. Fortunately, the relationship between discrete and continuous system is particularly strong for some subclasses. For example, in bounded EQ systems structural liveness as discrete and as continuous are equivalent [50].

Many works have been done related to control design of PWL systems [25, 26, 27]. But these control techniques can not be applied to contPN which is a particular case of PWL systems because of its particular characteristics, for example input constraints that depend on the state. Hence, there exists necessity for searching more control technique for this particular model.

3. A HEURISTICS FOR MINIMUM TIME CONTROL OF TIMED CONTINUOUS PETRI NETS

In this chapter we design a control strategy which drives a contPN from an initial state to target state in minimum time. The proposed strategy computes the control actions to drive the system through a linear trajectory. Then, as in [29], in order to obtain faster linear or piecewise linear trajectories, intermediate states, not necessarily on the line connecting the initial and the target marking, are computed. Moreover, an algorithm that recursively refines the initially obtained trajectory is given. These computations are then used to develop a closed loop scheme. A manufacturing system is taken as a case study and studied to obtain a number of products in minimum time. Many simulations were performed and some results are shown here.

3.1 Problem Statement

Our objective is to develop a method which computes a control action $\mathbf{u}$ that drives the system from the initial marking $\mathbf{m}_0$ to a desired target marking $\mathbf{m}_f$ through a linear trajectory. Such control actions are the result of solving a LPP. In order to improve the time to reach $\mathbf{m}_f$, developed method is used to compute linear or piecewise linear trajectories by introducing intermediate states which are calculated by means of BLP. Our procedure consists of assigning constant or piecewise constant controlled flow $\mathbf{w}$ that satisfies dynamic upper bounds.

Notice that in order to be reachable, $\mathbf{m}_f$ necessarily satisfies the state equation, i.e., $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}$. We assume that $\mathbf{m}_0$ and $\mathbf{m}_f$ are interior points of the reachability space, i.e., the markings are strictly positive. The assumption that $\mathbf{m}_0$ is positive ensures that the system can straightforwardly move at $\tau = 0$ in the direction of $\mathbf{m}_f$; the assumption that $\mathbf{m}_f$ is positive ensures that $\mathbf{m}_f$ can be reached [49] in finite time [31].

3.2 Controlled Timed Continuous Petri Nets

In contPN, a transition is associated in general to a machine and this machine can not work faster than its maximum firing rate, the only control action we consider is to brake it down. In other words, we assume that the only action that can be applied to contPN is to reduce the flow of transitions [23] If a transition can be controlled (its flow can be reduced or even stopped), we will say that it is a *controllable* transition [31]. In all control schemes developed in this thesis, it is assumed that all transitions are controllable.

Definition 3.1. *The controlled flow, $\mathbf{w}$, of a contPN is defined as $\mathbf{w}(\tau) = \mathbf{f}(\tau) - \mathbf{u}(\tau)$, with $0 \leq \mathbf{u}(\tau) \leq \mathbf{f}(\tau)$, where $\mathbf{f}$ is the flow of the uncontrolled system, i.e., defined as in 2.11, and $\mathbf{u}$ is the control action.*

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

$$\begin{aligned} \dot{\mathbf{m}} &= \mathbf{C} \cdot [\mathbf{f} - \mathbf{u}] = \mathbf{C} \cdot \mathbf{w} \\ 0 &\leq \mathbf{u} \leq \mathbf{f} \end{aligned} \tag{3.1}$$

The constraint $0 \leq \mathbf{u} \leq \mathbf{f}$ can be rewritten as $0 \leq \mathbf{f} - \mathbf{u} \leq \mathbf{f}$. From the Definition 3.1 $\mathbf{w} = \mathbf{f} - \mathbf{u}$ and using (2.10), the constraint can be expressed as:

$$0 \leq \mathbf{w} \leq \mathbf{\Lambda} \cdot \mathbf{\Pi}^{\eta(\mathbf{m})} \cdot \mathbf{m} \tag{3.2}$$

3.3 Computation of Linear Trajectories

In this section, we will distinguish two cases: $\mathbf{m}_0$ and $\mathbf{m}_f$ are in the same region (see Section 2.5.1) $\mathbf{m}_0$ and $\mathbf{m}_f$ are in different regions. In this case the crossing points of the straight line and borders must be considered.

3.3.1 $\mathbf{m}_0$ and $\mathbf{m}_f$ are in the same region

In the case that $\mathbf{m}_0$ and $\mathbf{m}_f$ are in the same region $\mathcal{R}^z$, i.e., $\eta(\mathbf{m}_f) = \eta(\mathbf{m}_0) = z$, then all the states on the straight line connecting them are also interior

points of $\mathcal{R}^z$ since all the regions are convex [48]. The programming problem (3.3) computes the correspondig control law:

$$\begin{aligned}
& \min_{\mathbf{w}} \quad \tau_f \\
& \text{s. t.} \quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{w} \cdot \tau_f \quad (a) \\
& \quad 0 \leq w_j \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min \{m_{0i}, m_{fi}\}, \\
& \quad \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji}^z \neq 0 \quad (b)
\end{aligned} \tag{3.3}$$

The equations correspond to: (a) the time dependent equation of the straight line connecting $\mathbf{m}_0$ to $\mathbf{m}_f$, (b) flow constraints in (3.2). Notice that (3.3)(b) is a linear constraint because m_{0i} and m_{fi} are known. The product $\mathbf{w} \cdot \tau_f$ makes (3.3) a BLP. But it can be transformed into a LPP by defining a new vector of variables $\mathbf{s} = \mathbf{w} \cdot \tau_f$. The resulting LPP is:

$$\begin{aligned}
& \min_{\mathbf{s}} \quad \tau_f \\
& \text{s. t.} \quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s} \quad (a) \\
& \quad 0 \leq s_j \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min \{m_{0i}, m_{fi}\} \cdot \tau_f, \\
& \quad \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji}^z \neq 0 \quad (b)
\end{aligned} \tag{3.4}$$

3.3.2 $\mathbf{m}_0$ and $\mathbf{m}_f$ are in different regions

In the case that $\mathbf{m}_0$ and $\mathbf{m}_f$ are in different regions, i.e., $\eta(\mathbf{m}_f) \neq \eta(\mathbf{m}_0)$, the line connecting $\mathbf{m}_0$ and $\mathbf{m}_f$ can be divided in several segments, each one starting in a border (or at $\mathbf{m}_0$ for the first segment) and ending in a border (or at $\mathbf{m}_f$ for the last one). Then LPP (3.4) is applied on each of these segments.

Algorithm 3.1 computes the total time τ_f by using LPP (3.4) for the linear trajectory ℓ from $\mathbf{m}_0$ to $\mathbf{m}_f$ which pass through $i \in \{1, 2, \dots, r+1\}$ regions. In this algorithm, $\mathbf{m}_c^i$ stands for state at the intersection of ℓ and the i^{th} crossed border $i \in \{1, 2, \dots, r\}$, i.e. border seperating regions $\mathcal{R}^i$ and $\mathcal{R}^{i+1}$. $\mathbf{m}_c^i$ is calculated by solving following equation system:

$$\begin{aligned}
& (\mathbf{\Pi}^i - \mathbf{\Pi}^{i+1}) \cdot \mathbf{m}_c^i = 0 \\
& \alpha \cdot \mathbf{m}_0 + (1 - \alpha) \cdot \mathbf{m}_f = 0
\end{aligned} \tag{3.5}$$

where $\mathbf{\Pi}^i$ is the constraint matrix of region $\mathcal{R}^i$. Let $\mathbf{m}_c^0$ and $\mathbf{m}_c^{r+1}$ denote $\mathbf{m}_0$ and $\mathbf{m}_f$, respectively and $\mathbf{w}^i$ denotes the flow obtained from the state $\mathbf{m}_c^i$ to

the state $\mathbf{m}_c^{i+1}$. Note that, if ℓ does not cross any border, then $r = 0$, that is $\mathbf{m}_c^0 = \mathbf{m}_0$ and $\mathbf{m}_c^1 = \mathbf{m}_f$.

Algorithm 3.1. Linear trajectory

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

Output: $\tau^1, \mathbf{w}^1, \tau^2, \mathbf{w}^2, \dots, \tau^{r+1}, \mathbf{w}^{r+1}, \tau_f = \sum_{i=1}^{r+1} \tau^i$

Compute the line ℓ connecting $\mathbf{m}_0$ and $\mathbf{m}_f$

Compute the intersection of ℓ and the crossed borders: $\mathbf{m}_c^1, \mathbf{m}_c^2, \dots, \mathbf{m}_c^r$

$\mathbf{m}_c^0 = \mathbf{m}_0, \mathbf{m}_c^{r+1} = \mathbf{m}_f$

for $i = 0$ to r **do**

Determine τ^i by solving LPP in (3.4) with $\mathbf{m}_0 = \mathbf{m}_c^i$ and $\mathbf{m}_f = \mathbf{m}_c^{i+1}$

end for

Proposition 3.1. Let $\langle \mathcal{N}, \boldsymbol{\lambda}, \mathbf{m}_0 \rangle$ be a contPN system with $\mathbf{m}_0 > 0$. If $\mathbf{m}_f$ belongs to $RS_{CL}^t(\mathcal{N}, \boldsymbol{\lambda}, \mathbf{m}_0)$ and $\mathbf{m}_f > 0$:

- LPP(3.4) is feasible.
- The control action given by Algorithm 3.1 ensures that $\mathbf{m}_f$ is reached in finite time.

Proof. Since $\mathbf{m}_f$ is a reachable marking, then there exists $\mathbf{s}$ such that the state equation, (3.4)(a), is satisfied. By taking τ_f sufficiently large, (3.4)(b) can be satisfied since $\lambda_j \cdot \Pi_{ji}^z \cdot \min \{m_{0i}, m_{fi}\} > 0$. Then, by [49] the linear trajectory from $\mathbf{m}_0$ to $\mathbf{m}_f$ can be followed by the system since $\mathbf{m}_0 > 0$ and $\mathbf{m}_f > 0$, i.e., all intermediate states are not spurious. Moreover, since $\mathbf{m}_f > 0$, by [31] $\mathbf{m}_f$ can be reached in finite time. $\square$

Example 3.1. Let us consider the control law design for the contPN in Figure 3.1. Assume $\boldsymbol{\lambda} = [4 \ 1 \ 3 \ 1]^T$, $\mathbf{m}_0 = [7.5 \ 4.5 \ 4 \ 2 \ 1.5 \ 5 \ 4 \ 2.5]^T$ and $\mathbf{m}_f = [7 \ 5 \ 5 \ 1 \ 1 \ 4 \ 5 \ 3]^T$.

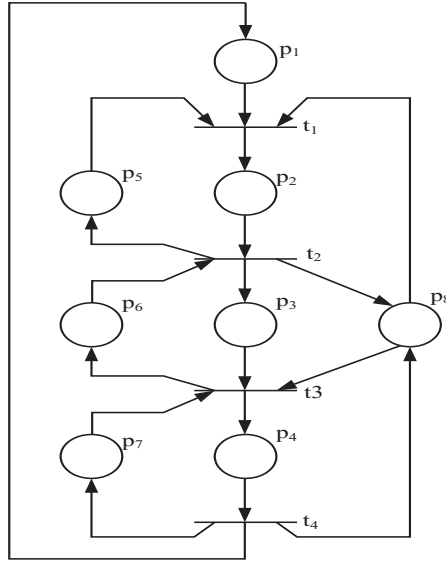


Figure 3.1: A PN taken from [3]

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

$$\begin{aligned}
 \dot{m}_1 &= f_4 - f_1 = \lambda_4 \cdot m_4 - \lambda_1 \cdot \min\{m_5, m_1, m_8\} \\
 \dot{m}_2 &= f_1 - f_2 = \lambda_1 \cdot \min\{m_5, m_1, m_8\} - \lambda_2 \cdot \min\{m_2, m_6\} \\
 \dot{m}_3 &= f_2 - f_3 = \lambda_2 \cdot \min\{m_2, m_6\} - \lambda_3 \cdot \min\{m_3, m_7, m_8\} \\
 \dot{m}_4 &= f_3 - f_4 = \lambda_3 \cdot \min\{m_3, m_7, m_8\} - \lambda_4 \cdot m_4 \\
 \dot{m}_5 &= f_2 - f_1 = \lambda_2 \cdot \min\{m_2, m_6\} - \lambda_1 \cdot \min\{m_5, m_1, m_8\} \\
 \dot{m}_6 &= f_3 - f_2 = \lambda_3 \cdot \min\{m_3, m_7, m_8\} - \lambda_2 \cdot \min\{m_2, m_6\} \\
 \dot{m}_7 &= f_4 - f_3 = \lambda_4 \cdot m_4 - \lambda_3 \cdot \min\{m_3, m_7, m_8\} \\
 \dot{m}_8 &= f_2 + f_4 - f_1 - f_3 = \lambda_2 \cdot \min\{m_2, m_6\} + \lambda_4 \cdot m_4 \\
 &\quad - \lambda_1 \cdot \min\{m_5, m_1, m_8\} - \lambda_3 \cdot \min\{m_3, m_7, m_8\}
 \end{aligned} \tag{3.6}$$

Note that the line ℓ connecting $\mathbf{m}_0$ and $\mathbf{m}_f$ crosses only one border: $m_2 = m_6$ and the crossing point is calculated as: $\mathbf{m}_c^1 = [7.33 \ 4.67 \ 4.33 \ 1.67 \ 1.33 \ 4.67 \ 4.33 \ 2.67]^T$, $\mathbf{m}_c^0 = \mathbf{m}_0$, $\mathbf{m}_c^2 = \mathbf{m}_f$. The trajectory is illustrated in Figure 3.2.

Note that $\mathbf{m}_0$ and $\mathbf{m}_f$ are in different regions:

- $\mathbf{m}_0 \in \mathcal{R}^1 : \{\mathbf{m} \mid m_5 \leq m_1, m_5 \leq m_8, m_8 \leq m_7, m_8 \leq m_3, m_2 \leq m_6\}$
- $\mathbf{m}_f \in \mathcal{R}^2 : \{\mathbf{m} \mid m_5 \leq m_1, m_5 \leq m_8, m_8 \leq m_7, m_8 \leq m_3, m_6 \leq m_2\}$.

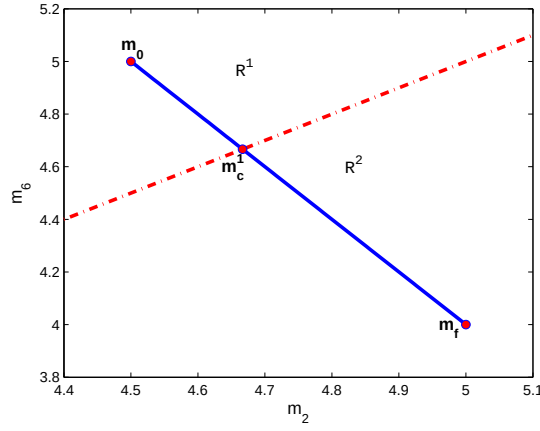


Figure 3.2: Trajectory for Example 3.1

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

$$\left\{ \begin{array}{l} \dot{m}_1 = m_4 - 4 \cdot m_5 \\ \dot{m}_2 = 4 \cdot m_5 - m_2 \\ \dot{m}_3 = m_2 - 3 \cdot m_8 \\ \dot{m}_4 = 3 \cdot m_8 - m_4 \\ \dot{m}_5 = m_2 - 4 \cdot m_5 \\ \dot{m}_6 = 3 \cdot m_8 - m_2 \\ \dot{m}_7 = m_4 - m_8 \\ \dot{m}_8 = m_2 + m_4 - 4 \cdot m_5 + 3 \cdot m_8 \end{array} \right. \quad (3.7)$$

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

$$\left\{ \begin{array}{l} \dot{m}_1 = m_4 - 4 \cdot m_5 \\ \dot{m}_2 = 4 \cdot m_5 - m_6 \\ \dot{m}_3 = m_6 - 3 \cdot m_8 \\ \dot{m}_4 = 3 \cdot m_8 - m_4 \\ \dot{m}_5 = m_6 - 4 \cdot m_5 \\ \dot{m}_6 = 3 \cdot m_8 - m_6 \\ \dot{m}_7 = m_4 - m_8 \\ \dot{m}_8 = m_6 + m_4 - 4 \cdot m_5 + 3 \cdot m_8 \end{array} \right. \quad (3.8)$$

Let us consider the control law design for the considered contPN. According to Algorithm 3.1, first LPP (3.4) with $\mathbf{m}_0 = \mathbf{m}_c^0$ and $\mathbf{m}_f = \mathbf{m}_c^1$ is solved. Its constraints are:

$$\begin{aligned}
7.33 &= s_4 - s_1 + 7.5 \\
4.67 &= s_1 - s_2 + 4.5 \\
4.33 &= s_2 - s_3 + 4 \\
1.67 &= s_3 - s_4 + 2 \\
1.33 &= s_2 - s_1 + 1.5 \\
4.67 &= s_3 - s_2 + 5 \\
4.33 &= s_4 - s_3 + 4 \\
2.67 &= s_2 + s_4 - s_1 - s_3 + 2.5
\end{aligned} \tag{a}$$

$$\begin{aligned}
0 &\leq s_1 \leq 4 \cdot 1.33 \cdot \tau_f \\
0 &\leq s_2 \leq 4.5 \cdot \tau_f \\
0 &\leq s_3 \leq 3 \cdot 2.5 \tau_f \\
0 &\leq s_4 \leq 1.66 \cdot \tau_f
\end{aligned} \tag{b}$$

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

$$\begin{aligned}
7 &= s_4 - s_1 + 7.33 \\
5 &= s_1 - s_2 + 4.67 \\
5 &= s_2 - s_3 + 4.33 \\
1 &= s_3 - s_4 + 1.67 \\
1 &= s_2 - s_1 + 1.33 \\
4 &= s_3 - s_2 + 4.67 \\
5 &= s_4 - s_3 + 4.33 \\
3 &= s_2 + s_4 - s_1 - s_3 + 2.67
\end{aligned} \tag{a}$$

$$\begin{aligned}
0 &\leq s_1 \leq 4 \cdot \tau_f \\
0 &\leq s_2 \leq 4 \cdot \tau_f \\
0 &\leq s_3 \leq 3 \cdot 2.67 \cdot \tau_f \\
0 &\leq s_4 \leq \tau_f
\end{aligned} \tag{b}$$

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

$$\mathbf{u}(\tau) = \begin{cases} \begin{bmatrix} 4 \cdot m_5(\tau) - 2.49 \\ m_2(\tau) - 1.67 \\ 3 \cdot m_8(\tau) \\ m_4(\tau) - 1.67 \end{bmatrix}, & \text{if } 0 \leq \tau \leq 0.2 \\ \begin{bmatrix} 4 \cdot m_5(\tau) - 1.5 \\ m_2(\tau) - 1 \\ 3 \cdot m_8(\tau) \\ m_4(\tau) - 1 \end{bmatrix}, & \text{if } 0.2 < \tau \leq 0.87 \end{cases} \quad (3.9)$$

Figure 3.3(a) illustrates the convergence of the markings $\mathbf{m}$ under the designed control law, while Figure 3.4(b) shows the control signal $\mathbf{u}$, $\mathbf{w}$ and their upper bound.

3.4 Reducing the time by computing a piecewise linear trajectory

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

The designed control in the previous subsection takes the system from $\mathbf{m}_0$ to $\mathbf{m}_f$ through a linear trajectory by minimizing the time duration. In order to improve the time spent to move from $\mathbf{m}_0$ to $\mathbf{m}_f$, intermediate states denoted by $\mathbf{m}^k$ where $k \in \{1, 2, \dots, r\}$, that do not necessarily lie on the line from $\mathbf{m}_0$ to $\mathbf{m}_f$, are computed. The new trajectory is obtained as the union of $r + 1$ segments: $\mathbf{m}_0 \rightarrow \mathbf{m}^1 \rightarrow \mathbf{m}^2 \rightarrow \dots \rightarrow \mathbf{m}^r \rightarrow \mathbf{m}_f$. In this work, we consider the intermediate states in the range: $\mathcal{R}^I = \{\mathbf{m} \mid m_i \leq \max\{m_{f_i}, m_{0_i}\}, m_i \geq \min\{m_{f_i}, m_{0_i}\}, i \in \{1, 2, \dots, |P|\}\}$. Because $\mathbf{m}_0, \mathbf{m}_f > 0$, all $\mathbf{m} \in \mathcal{R}^I$ are positive and reachable [49].

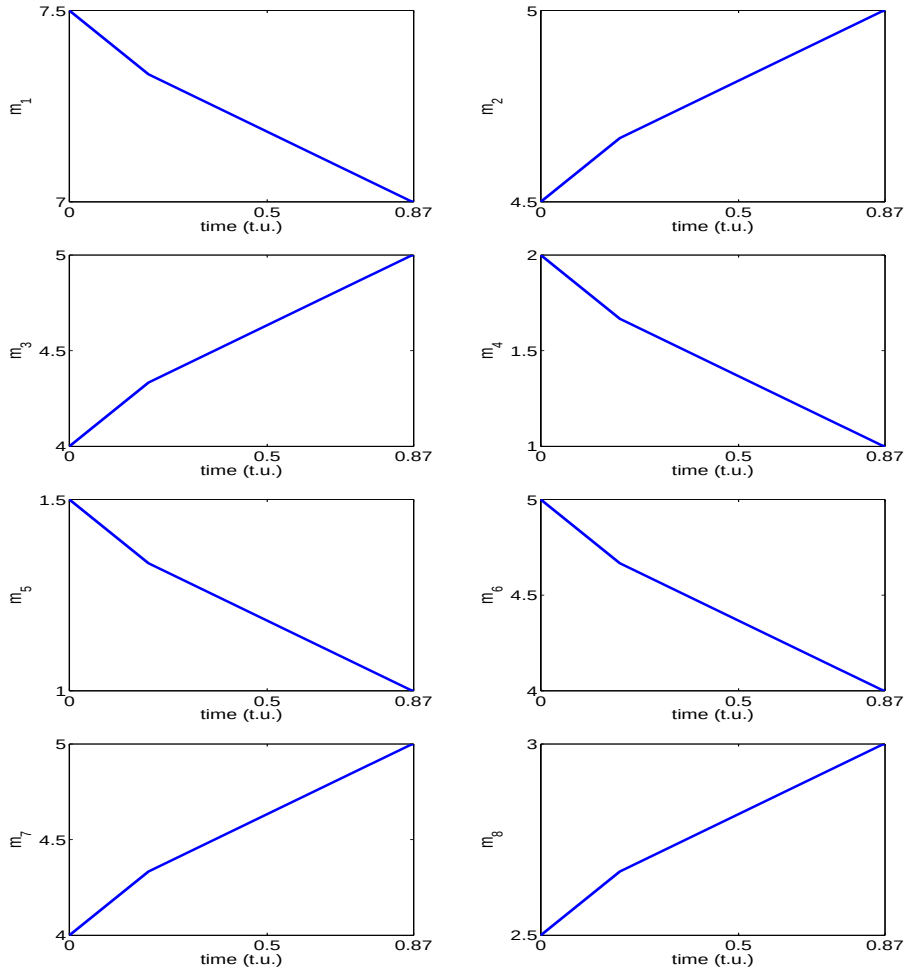


Figure 3.3: Evolution of markings for Example 3.1

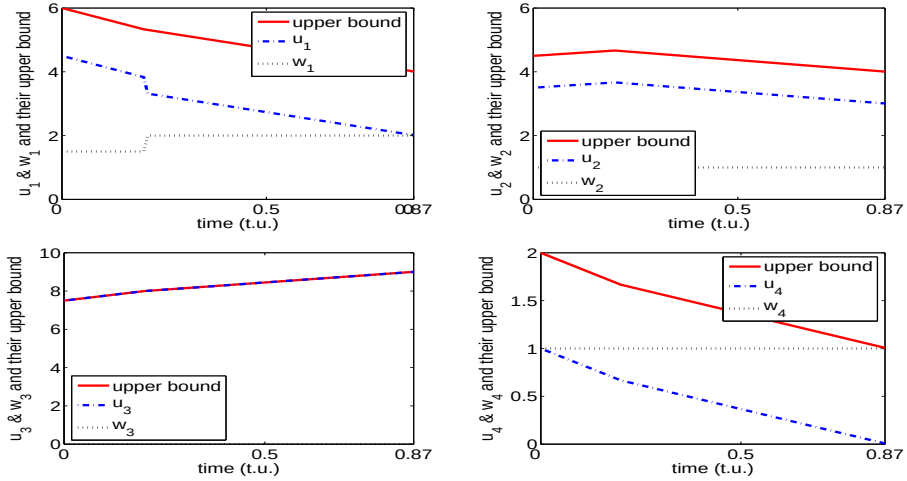


Figure 3.4: Evolution of control inputs, controlled flows and their upper bounds for Example 3.1

3.4.1 Intermediate states on the borders

Let the line from $\mathbf{m}_0$ to $\mathbf{m}_f$ cross r borders, and pass through $r+1$ different regions: $\mathcal{R}^1, \mathcal{R}^2, \dots, \mathcal{R}^{r+1}$ with corresponding constraint matrices $\mathbf{\Pi}^1, \mathbf{\Pi}^2 \dots \mathbf{\Pi}^{r+1}$. We assign constant flows for each transitions during each line segments and the intermediate states on each border: $\mathbf{m}^1, \mathbf{m}^2, \dots, \mathbf{m}^r$. Under these assumptions, BLP (3.10), at which $\mathbf{m}^0 = \mathbf{m}_0$ $\mathbf{m}^{r+1} = \mathbf{m}_f$ and $\mathbf{s}^k = \mathbf{w}^k \cdot \tau^k$ $k \in \{1, \dots, r+1\}$, obtains the intermediate states that yield a minimum time trajectory.

Variables: $\mathbf{m}^1, \mathbf{m}^2, \dots, \mathbf{m}^k$ $k \in \{1, \dots, r\}$ and $\mathbf{s}^k, \tau^k$ $k \in \{1, \dots, r+1\}$

$$\min \sum_{k=1}^{s+1} \tau^k$$

$$\text{s. t.} \quad \mathbf{m}^{k+1} = \mathbf{m}^k + \mathbf{C} \cdot \mathbf{s}^{k+1}, \quad k \in \{0, 1, 2, \dots, r\} \quad (a)$$

$$\left(\mathbf{\Pi}^k - \mathbf{\Pi}^{k+1} \right) \cdot \mathbf{m}^k = 0, \quad k \in \{1, 2, \dots, r\} \quad (b)$$

$$m_i^k \leq m_i^{k+1} \quad \text{if } m_{0_i} \leq m_{f_i} \\ i \in \{1, 2, \dots, |P|\}, \quad k \in \{0, 1, 2, \dots, r\} \quad (c)$$

$$m_i^k \geq m_i^{k+1} \quad \text{if } m_{0_i} \geq m_{f_i} \\ i \in \{1, 2, \dots, |P|\}, \quad k \in \{0, 1, 2, \dots, r\} \quad (d)$$

$$0 \leq s_j^k \leq \lambda_j \cdot \Pi_{ji}^k \cdot \min\{m_i^{k-1}, m_i^k\} \cdot \tau^k, \\ \text{with } p_i \text{ st. } \Pi_{ji}^k \neq 0, j \in \{1, 2, \dots, |T|\}, \quad k \in \{1, 2, \dots, r\} \quad (e) \\ (3.10)$$

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

3.4.2 Interior intermediate states

In the case that $\mathbf{m}_0$ and $\mathbf{m}_f$ are in the same region $\mathcal{R}^z$ a recursive method can be used to determine intermediate states. The same recursive method can also

be applied to find additional intermediate states, $\mathbf{m}^d$, between $\mathbf{m}^k$ and $\mathbf{m}^{k+1}$. In this method we keep computing intermediate states until the time cannot be improved by a considerable amount. In this case, BLP that will be solved in each step can be simplified to BLP (3.11) where $\mathbf{s}^1 = \mathbf{w}^1 \cdot \tau^1$, $\mathbf{s}^2 = \mathbf{w}^2 \cdot \tau^2$.

Variables: $\mathbf{m}^d, \mathbf{s}^1, \mathbf{s}^2, \tau^1, \tau^2, \sigma$

$$\min \quad \tau^1 + \tau^2$$

$$\text{s. t.} \quad \mathbf{m}^d = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s}^1 \quad (a_1)$$

$$\mathbf{m}_f = \mathbf{m}^d + \mathbf{C} \cdot \mathbf{s}^2, \quad (a_2)$$

$$\min\{m_{f_i}, m_{0_i}\} \leq m_i^d \leq \max\{m_{f_i}, m_{0_i}\}, \quad (3.11)$$

$$\forall i \in \{1, 2, \dots, |P|\} \quad (b)$$

$$0 \leq s_j^1 \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min\{m_{0_i}, m_i^d\} \cdot \tau^1,$$

$$\text{with } i \text{ st. } \Pi_{ji}^k \neq 0, \quad j \in \{1, 2, \dots, |T|\} \quad (c_1)$$

$$0 \leq s_j^2 \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min\{m_i^d, m_{f_i}\} \cdot \tau^2,$$

$$\text{with } i \text{ st. } \Pi_{ji}^k \neq 0, \quad j \in \{1, 2, \dots, |T|\} \quad (c_2)$$

The constraints correspond to (a_1) the equation of the line connecting $\mathbf{m}_0$ to $\mathbf{m}^d$: $\mathbf{m}^d = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{w}^1 \cdot \tau^1$ and $\mathbf{m}^d$ is a reachable marking (a_2) the time dependent equation of the line connecting $\mathbf{m}^d$ to $\mathbf{m}_f$: $\mathbf{m}_f = \mathbf{m}^d + \mathbf{C} \cdot \mathbf{w}^2 \cdot \tau^2$; (c_1) the constraints on the controlled flow $\mathbf{w}^1$; (c_2) the constraints on the controlled flow $\mathbf{w}^2$.

Algorithm 3.2 expresses the recursive method we proposed. It computes new intermediate states until the stopping criterion is satisfied. It is satisfied when the time of the trajectory cannot be decreased more than a given value ϵ . In the algorithm, *path* is a global variable storing the ordered set of couples of states in the trajectory and the relative times to reach them. The number of states in *path* is denoted by $|path|$ and the i^{th} element of the set *path* is denoted by $path(i)$.

Algorithm 3.2. Piecewise Linear Trajectory

Input: $\langle \mathcal{N}, \mathbf{m}_0 \rangle, \mathbf{m}_f, \epsilon$

Output: $path$

Global variable: $path$

Compute the line ℓ connecting $\mathbf{m}_0$ and $\mathbf{m}_f$

Determine the number of crossed borders: r

if $r = 0$ **then**

Solve (3.4) to obtain τ_f

$path_0 = \{(\mathbf{m}_0, 0), (\mathbf{m}_f, \tau_f)\}$

else

Solve (3.10) to obtain τ_f

$path_0 = \{(\mathbf{m}_0, 0), (\mathbf{m}^1, \tau^1), \dots, (\mathbf{m}_f, \tau_f)\}$

end if

$\tau^x = \tau_f$, $path = path_0$

for $i = 1 : |path_0| - 1$ **do**

$(\mathbf{m}^x, \tau^x) = path_0(i)$, $(\mathbf{m}^y, \tau^y) = path_0(i + 1)$

Call Split function with $((\mathbf{m}^x, \tau^x), (\mathbf{m}^y, \tau^y))$

end for

Calculate the total time for $path$, i.e., τ^y

Split function

Input: $(\mathbf{m}^x, \tau^x), (\mathbf{m}^y, \tau^y)$

Output: $path$

Solve (3.11) to obtain $\tau^1, \tau^2, \mathbf{m}^d$

if $\tau^y - (\tau^1 + \tau^2) > \epsilon$ **then**

Insert $(\mathbf{m}^d, \tau^1)$ in $path$

Change $(\mathbf{m}^y, \tau^y)$ in $path$ by $(\mathbf{m}^y, \tau^2)$

Call Split function with $((\mathbf{m}^x, \tau^x), (\mathbf{m}^d, \tau^1))$

Call Split function with $((\mathbf{m}^d, \tau^1), (\mathbf{m}^y, \tau^2))$

end if

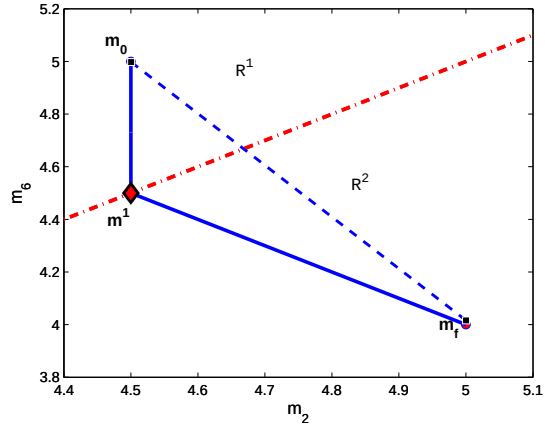
Example 3.2. Let us consider the contPN system in Figure 3.1 again. Since the linear trajectory between $\mathbf{m}_0$ and $\mathbf{m}_f$ crosses only one border, $r = 1$ and solving BLP (3.10) with $r = 1$ yields 0.83 t.u. as a total time. And the new path is $\mathbf{m}_0 \rightarrow \mathbf{m}^1 \rightarrow \mathbf{m}_f$, where $\mathbf{m}^1 = [7.5 \ 4.5 \ 4.5 \ 1.5 \ 1.5 \ 4.5 \ 4.5 \ 3]^T$. For an additional intermediate state, say $\mathbf{m}'^1$, between $\mathbf{m}_0$ and $\mathbf{m}^1$, we solve (3.11) for $\mathbf{m}_0 = [7.5 \ 4.5 \ 4 \ 2 \ 1.5 \ 5 \ 4 \ 2.5]^T$ and $\mathbf{m}_f = \mathbf{m}^1$. Similarly, for an additional intermediate state between $\mathbf{m}^1$ and $\mathbf{m}_f$, say $\mathbf{m}'^2$, we solve (3.11) for $\mathbf{m}_0 = \mathbf{m}^1$ and $\mathbf{m}_f = [7 \ 5 \ 5 \ 1 \ 1 \ 4 \ 5 \ 3]^T$. And the new path $\mathbf{m}_0 \rightarrow \mathbf{m}'^1 \rightarrow \mathbf{m}^1 \rightarrow \mathbf{m}'^2 \rightarrow \mathbf{m}_f$ is obtained, where $\mathbf{m}'^1 = [7.5 \ 4.5 \ 4.26 \ 1.74 \ 1.5 \ 4.74 \ 4.26 \ 2.76]^T$ and $\mathbf{m}'^2 = [7.45 \ 4.78 \ 4.55 \ 1.22 \ 1.22 \ 4.45 \ 4.78 \ 3]^T$ (For the Matlab code see Appendix A.2). The total time is reduced to 0.74 t.u. with the control action $u(\tau) = [4 \cdot m_5(\tau) - 1.73 \ m_2(\tau) - 1.73 \ 3 \cdot m_8(\tau) \ m_4(\tau) - 1.7321]^T$ for $0 \leq \tau \leq 0.15$; $u(\tau) = [4 \cdot m_5(\tau) - 1.5 \ m_2(\tau) - 1.5 \ 3 \cdot m_8(\tau) \ m_4(\tau) - 1.5]^T$ for $0.15 \leq \tau \leq 0.3$; $u(\tau) = [4 \cdot m_5(\tau) - 1.23 \ m_6(\tau) \ 3 \cdot m_8(\tau) \ m_4(\tau) - 1.2247]^T$ for $0.3 \leq \tau \leq 0.51$; $u(\tau) = [4 \cdot m_5(\tau) - 3.21 \ m_6(\tau) - 2.21 \ 3 \cdot m_8(\tau) \ m_4(\tau) - 1]^T$ for $0.51 \leq \tau \leq 0.74$. The trajectories for one and three intermediate states are illustrated in Figure 3.5(a) and Figure 3.5(b), respectively.

Figure 3.6(a) illustrates the convergence of the markings $\mathbf{m}$ under the designed control law, while Figure 3.7(b) shows the control inputs $\mathbf{u}$, and the controlled flows $\mathbf{w}$, and their upper bound. The total time duration for several intermediate states can be found in Table 3.1.

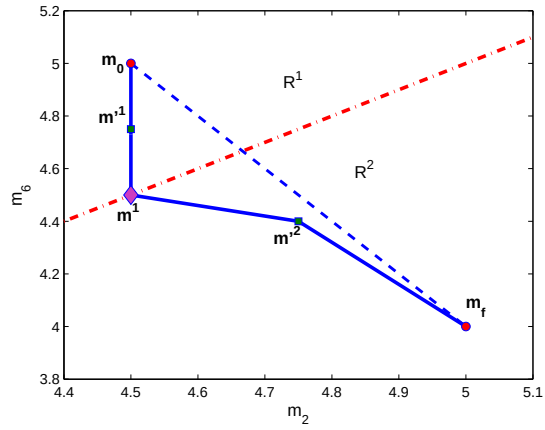
Table 3.1: The total time duration of intermediate states for Example 3.2

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

Example 3.3. Let us consider the net in Figure 2.4 with $\boldsymbol{\lambda} = [3 \ 1 \ 1]^T$. By using Algorithm 3.1, the total time to reach $\mathbf{m}_f = [1 \ 10 \ 6 \ 2]^T$ from $\mathbf{m}_0 = [13 \ 3 \ 1 \ 10]^T$ through a linear trajectory is obtained as 2.43 t.u., with the control $u(\tau) = [3 \cdot m_4(\tau) - 7 \ m_2(\tau) \ m_3(\tau) - 1]^T$ for $0 \leq \tau \leq 0.93$, $u(\tau) = [3 \cdot$



(a)



(b)

Figure 3.5: Trajectory for (a)1 intermediate state and (b)3 intermediate states for Example 3.2

$m_4(\tau) - 12 \quad m_4(\tau) \quad m_3(\tau) - 1.71]^T$ for $0.93 \leq \tau \leq 1.26$, $u(\tau) = [3 \cdot m_1(\tau) - 3 \quad m_2(\tau) \quad m_3(\tau) - 0.42]^T$ for $1.26 \leq \tau \leq 2.43$, while $\mathbf{m}_f$ was reached in 4.4 t.u. in ([29]). The piecewise linear trajectory with 4 intermediate states obtained by Algorithm 2 yields 1.38 t.u., while $\mathbf{m}_f$ was reached in 4.4 t.u. in [29]. The piecewise linear trajectory with 4 intermediate states obtained by Algorithm 3.2 yields 1.38 t.u.

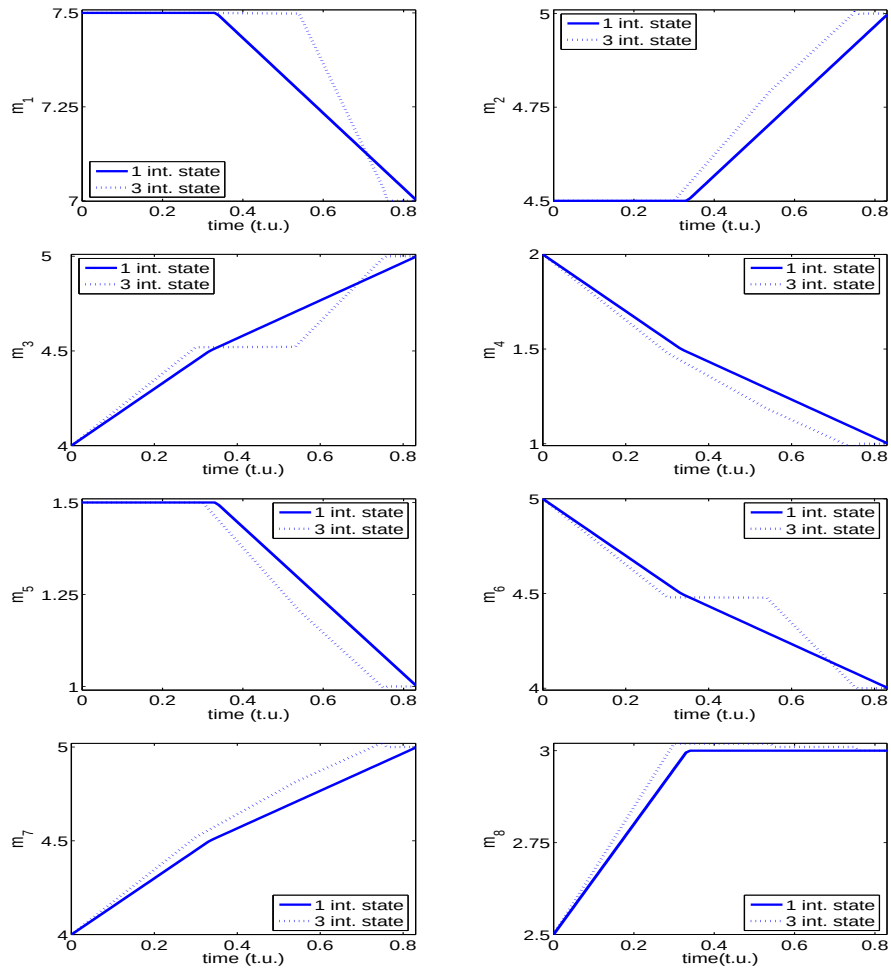


Figure 3.6: Evolution of markings for Example 3.2

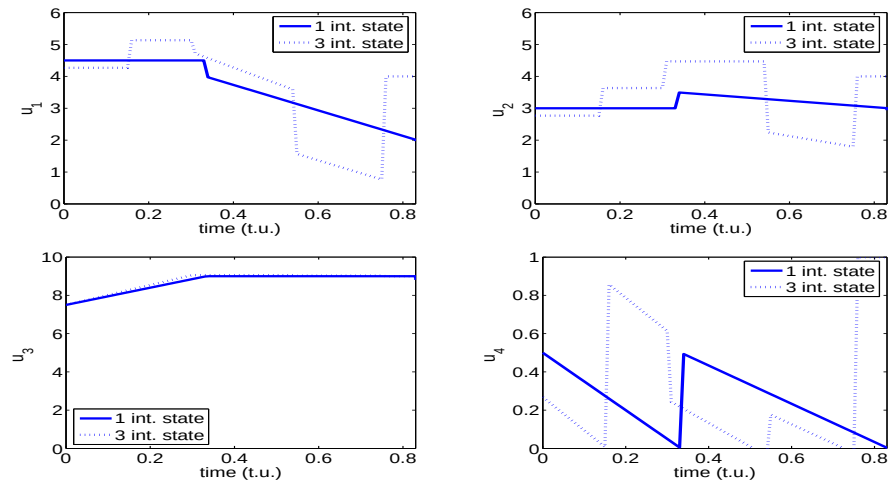


Figure 3.7: Evolution of control inputs for Example 3.2

3.5 Closed Loop Control

Using Algorithm 3.2 we obtain an offline controller. Obviously, in a real system, perturbations affect the evolution and when the control law is implemented, the noise effect should be eliminated. In this section, we assume that the PWL trajectory $path = \{\mathbf{m}_0, \mathbf{m}'^1, \dots, \mathbf{m}'^s, \mathbf{m}_f\}$ has been computed using Algorithm 3.2 and our problem is to follow this trajectory online. For the online implementation we consider the discrete-time representation of the system [28].

The discrete-time representation of the continuous-time system (3.1) is given by:

$$\begin{aligned} \mathbf{m}[k+1] &= \mathbf{m}[k] + \Theta \cdot \mathbf{C} \cdot \mathbf{w}[k] \\ 0 \leq \mathbf{w}[k] &\leq \mathbf{\Lambda} \cdot \mathbf{\Pi}^\eta(\mathbf{m}[k]) \cdot \mathbf{m}[k] \end{aligned} \quad (3.12)$$

Here Θ is the sampling period ($\tau = k \cdot \Theta$) and $\mathbf{m}[k]$ is the marking at step k , i.e., at time $k \cdot \Theta$. The sampling period should be small enough to avoid to reach negative markings. Let us consider a place p_i with $p_i^\bullet = \{t_1, t_2, \dots, t_j\}$ and $m[k]_i > 0$. Then $m[k+1]_i = m[k]_i + \Theta \cdot \mathbf{C}(i, :) \cdot \mathbf{w}[k] \geq m[k]_i - \Theta \cdot (\lambda_1 + \lambda_2 + \dots + \lambda_j) \cdot m[k]_i = m[k]_i \cdot (1 - \sum_{t_j \in p^\bullet} \lambda_j \cdot \Theta) \geq 0$. Hence, if Θ is chosen such that

$$\sum_{t_j \in p^\bullet} \lambda_j \cdot \Theta < 1 \quad (3.13)$$

then any marking reachable from $\mathbf{m}_0 = \mathbf{m}[0] \geq 0$ is nonnegative [28].

Our approach will be: for each segment of trajectory we will apply during each discrete-step $k \cdot \Theta$ the maximum possible flow solving a LPP. In fact, the procedure can be seen as a RHC scheme with timing horizon 1 and having as optimization index the minimization of the time. In [28] it is proved that using this control scheme the closed loop system is asymptotically stable.

In order to design the closed loop control for $path$ (see output of Algorithm 3.2), Algorithm 3.3 is developed. In the algorithm, the procedure is applied to each segment $(\mathbf{m}^j, \mathbf{m}^{j+1})$ of the $path$. Therefore, the first problem will be to reach $\mathbf{m}^1$ from $\mathbf{m}_0$. In order to do this, LPP (3.14) computes at $k \cdot \Theta$ the maximum distance that can be executed from the actual marking, $\mathbf{m}[k]$, in the direction of $\mathbf{m}^{j+1}$ during the next Θ t.u. This LPP tries to maximize

the distance during the next Θ that can be executed from the actual marking (constraint 1) such that the obtained intermediate marking $\mathbf{m}_{next}$ belongs to the straight line from the actual marking $\mathbf{m}[k]$ to the final one $\mathbf{m}^{j+1}$ (constraints 2 and 3) while the controlled flow is dynamically bounded (constraint 4). Then the same procedure is applied when the initial marking is $\mathbf{m}^1$ and $\mathbf{m}^2$ should be reached and so on. The procedure stops when a marking sufficiently close to $\mathbf{m}_f$ is reached. We will denote the noise that affects the flow by $\mathbf{z}$ (see equation (3.15)).

Algorithm 3.3. Closed Loop Control

Input: $\langle \mathcal{N}, \mathbf{m}_0 \rangle, \mathbf{m}_f, path, r$

$\mathbf{m}^0 = \mathbf{m}_0; \mathbf{m}^{r+1} = \mathbf{m}_f; \mathbf{m}[0] = \mathbf{m}^0; k = 0;$

for $j = 0 : r$ **do**

while $\|\mathbf{m}[k] - \mathbf{m}^{j+1}\| > \epsilon$ **do**

 Solve the following LPP

$$\begin{aligned}
 &\text{Variables: } \mathbf{w}, \alpha, \mathbf{m}'_{next} \\
 &\max \quad \alpha \\
 &\text{s.t.} \quad \mathbf{m}'_{next} = \mathbf{m}[k] + \Theta \cdot \mathbf{C} \cdot \mathbf{w} \\
 &\quad \mathbf{m}'_{next} = (1 - \alpha) \cdot \mathbf{m}[k] + \alpha \cdot \mathbf{m}^{j+1} \\
 &\quad 0 \leq \alpha \leq 1 \\
 &\quad 0 \leq \mathbf{w} \leq \mathbf{\Lambda} \cdot \mathbf{\Pi}^{\eta(\mathbf{m}[k])} \cdot \min\{\mathbf{m}[k], \mathbf{m}'_{next}\}
 \end{aligned} \tag{3.14}$$

 Apply the control to the system during Θ t.u. and measure $\mathbf{m}[k+1]$:

$$\mathbf{m}[k+1] = \mathbf{m}[k] + \Theta \cdot \mathbf{C} \cdot (\mathbf{w} + \mathbf{z}) \tag{3.15}$$

$k = k + 1$

end while

end for

If there is no perturbation, the total time to reach the desired marking by closed loop procedure can be smaller than the time obtained by the open loop procedure. This is due to the fact that the flow in the open loop control

that drive the system from a marking $\mathbf{m}^j$ to a marking $\mathbf{m}^{j+1}$ is constant and it is bounded by these two markings. In the closed loop implementation, the flow at step k is bounded by the actual marking and $\mathbf{m}^{j+1}$. Obviously, being the actual marking a convex combination of $\mathbf{m}^j$ and $\mathbf{m}^{j+1}$ the flow may be greater. On the other hand, because of discrete time implementation, the time of the closed loop control can be greater in some cases. For example, let us assume that the optimal time computed by the offline controller to go from $\mathbf{m}^j$ to $\mathbf{m}^{j+1}$ is 0.3 t.u. Taking a sampling time equal to 0.25 and assuming a maximum flow during trajectory equal to 1, in the absence of any perturbation, at least two steps are required to reach the desired marking. This corresponds to, at least, 0.5 t.u. that is greater than the open loop computation.

Example 3.4. *Let us consider the contPN system considered in Example 3.1 with the same initial and target marking $\mathbf{m}_0 = [7.5 \ 4.5 \ 4 \ 2 \ 1.5 \ 5 \ 4 \ 2.5]^T$ and $\mathbf{m}_f = [7 \ 5 \ 5 \ 1 \ 1 \ 4 \ 5 \ 3]^T$. In Example 3.2, path was obtained by using Algorithm 3.2 as $\text{path} = \{\mathbf{m}_0, \mathbf{m}^1, \mathbf{m}^2, \mathbf{m}^3, \mathbf{m}_f\}$ where $\mathbf{m}^1 = [7.5 \ 4.5 \ 4.26 \ 1.74 \ 1.5 \ 4.74 \ 4.26 \ 2.76]^T$, $\mathbf{m}^2 = [7.5 \ 4.5 \ 4.5 \ 1.5 \ 1.5 \ 4.5 \ 4.5 \ 3]^T$ and $\mathbf{m}^3 = [7.45 \ 4.78 \ 4.55 \ 1.22 \ 1.22 \ 4.45 \ 4.78 \ 3]^T$. According to inequality (3.13), $\Theta \leq 1/7$ must be satisfied. Let us assume that $\Theta = 0.01$ and the noise effect is zero.*

According to Algorithm 3.3, first we consider the first segment $\mathbf{m}_0 - \mathbf{m}^1$. That is, LPP (3.14) is solved at $k = 0$ to calculate the maximum distance that can be executed from $\mathbf{m}_0$ during Θ t.u in the direction of $\mathbf{m}^1$. This yields $\mathbf{w} = [2.02 \ 2.02 \ 0 \ 2.02]^T$. Applying corresponding control to the system during Θ t.u. drives the system to $\mathbf{m}[1] = [7.5 \ 4.5 \ 4.02 \ 1.98 \ 1.5 \ 4.98 \ 4.02 \ 2.52]^T$.

Since $\|\mathbf{m}[1] - \mathbf{m}^1\| > \epsilon$, we solve LPP(3.14) considering $\mathbf{m}[1]$ the initial marking. which yields $\mathbf{w} = [1.99 \ 1.99 \ 0 \ 1.99]^T$. Applying corresponding control to the system during Θ t.u. drives the system to $\mathbf{m}[2] = [7.5 \ 4.5 \ 4.04 \ 1.96 \ 1.5 \ 4.96 \ 4.04 \ 2.54]^T$ (For the Matlab code see Appendix A.3).

As a result, by applying the closed loop control approach given by Algorithm 3.3, $\mathbf{m}_f$ is reached from $\mathbf{m}_0$ by 66 time steps which correspond 0.66 t.u. Evolution of markings and control inputs of this system are given in Figure 3.8 and 3.9, respectively.

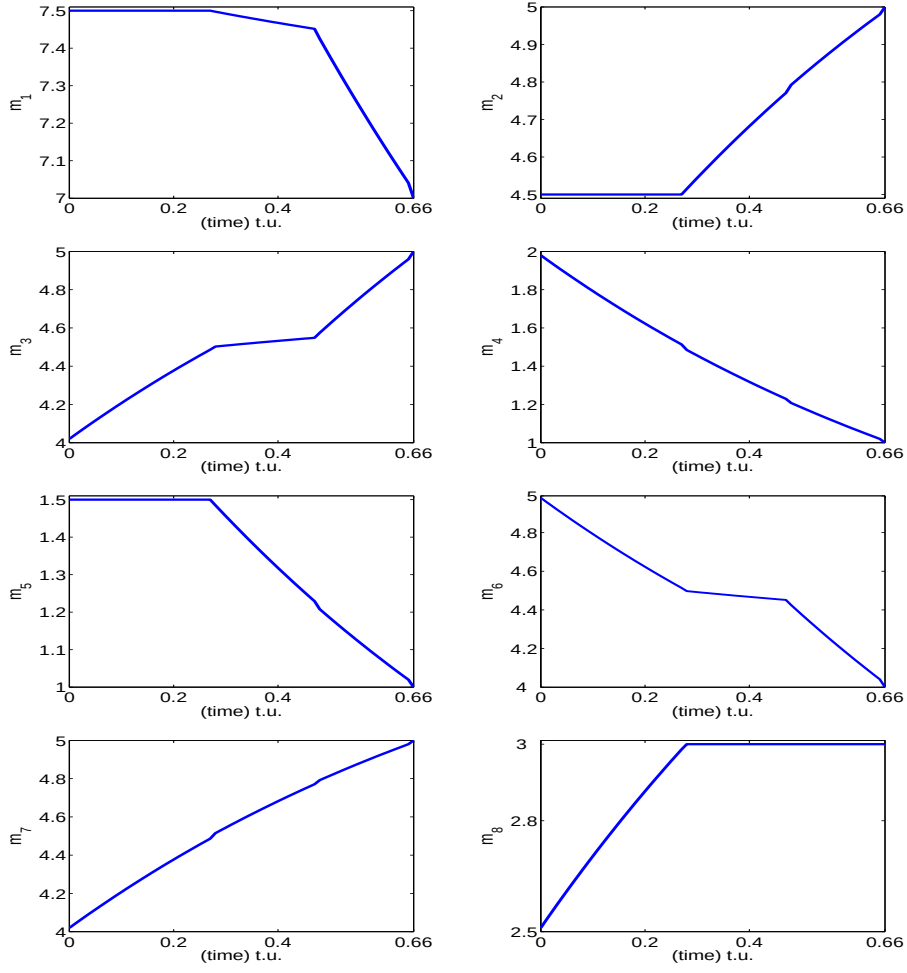


Figure 3.8: Evolution of markings for Example 3.4

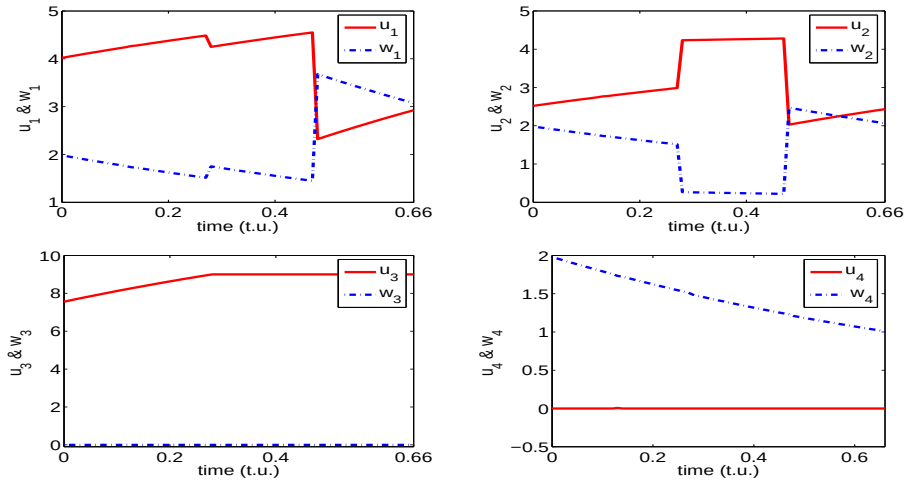


Figure 3.9: Evolution of control inputs for Example 3.4

Comparison of total durations and CPU times for open and closed control scheme is given in Table 3.2.

Table 3.2: Comparison of control schemes for Example 3.4

Control scheme	open loop control	closed loop control
Total duration (t.u.)	0.74	0.66
CPU time (sec)	0.32	14.1

3.6 Case study

In this section, we will apply our heuristics method for minimum time control to a manufacturing system. Let us consider again the net in Figure 2.5. This net produces final product by composing two different parts, A and B . The system has 3 sequential machines (M_1, M_2, M_3), 4 intermediate deposits (B_1A, B_1B, B_2A, B_2B) and one terminal deposit (B_3).

Let us assume that initially, there are 35 pallets of type A ($\mathbf{m}_0[Pallets_A] = 35$); 25 pallets of type B ($\mathbf{m}_0[Pallets_B] = 25$); one machine of each type from which the first two are initially in *Idle* state, while the third one is in *working* state ($\mathbf{m}_0[M1_Idle] = \mathbf{m}_0[M2_Idle] = \mathbf{m}_0[M3_Work] = 1$); one final product of type A and one of type B produced ($\mathbf{m}_0[B_2A] = \mathbf{m}_0[B_2B] = 1$); a maximum of 10 parts of type A and B can be produced at each time ($\mathbf{m}_0[Max_A] = \mathbf{m}_0[Max_B] = 10$); and finally, the capacity of the buffer to store the final products just before the recycling of pallets is equal to 25 ($\mathbf{m}_0[B_3_Empty] = 25$). Let us consider the task of producing 10 final product as fast as possible, i.e., in minimum time. Let the final marking for this control goal be: $\mathbf{m}_f[Pallets_A] = 27$, $\mathbf{m}_f[Pallets_B] = 17$, $\mathbf{m}_f[Max_A] = \mathbf{m}_f[Max_B] = 10$, $\mathbf{m}_f[M1_Idle] = \mathbf{m}_f[M2_Idle] = \mathbf{m}_f[M3_Idle] = 1$, $\mathbf{m}_f[B_3_Empty] = 15$ and $\mathbf{m}_f[B_3] = 10$. As previously established, it is assumed that the initial and target markings are strictly positive, i.e. , $m_0(p), m_f(p) \geq \mu$. In this example $\mu = 0.01$. Driving the system from $\mathbf{m}_0$ to $\mathbf{m}_f$ through the linear trajectory (see LPP (3.4)) by assigning constant flows

Table 3.3: Intermediate States for the case study

place	$\mathbf{m}_0$	$\mathbf{m}^1$	$\mathbf{m}^2$	$\mathbf{m}^3$	$\mathbf{m}^4$	$\mathbf{m}_f$
Pallet_A	35.0100	33.5000	31.0000	30.0000	28.5008	27.0000
M1_A	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
B1_A	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
M2_A	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
Max_A	10.0100	10.0000	10.0000	10.0000	10.0000	10.0000
M1_IDLE	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
M2_IDLE	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Pallet_B	25.0100	23.5000	21.0000	20.0000	18.5008	17.0000
M2_B	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
B1_B	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
M1_B	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
Max_B	10.0100	10.0000	10.0000	10.0000	10.0000	10.0000
B2_A	1.0100	1.0100	0.5100	0.0149	0.0100	0.0100
B2_B	1.0100	1.0100	0.5100	0.0149	0.0100	0.0100
M3_Idle	0.0100	0.5100	0.5100	1.0001	1.0099	1.0100
M3_Work	1.0100	0.5100	0.5100	0.0199	0.0101	0.0100
B3_Empty	25.0100	23.0000	20.0000	18.0148	16.5009	15.0000
B_3	0.0100	2.0100	5.0100	6.9952	8.5091	10.0100

to transitions takes 1000 t.u.

In order to improve the time and refine the trajectory, we apply Algorithm 3.2 with $\epsilon = 0.5$. The resulting path is composed by 4 intermediate states. Therefore 5 line segments, i.e., $path = \{\mathbf{m}_0 \rightarrow \mathbf{m}'^1 \rightarrow \mathbf{m}'^2 \rightarrow \mathbf{m}'^3 \rightarrow \mathbf{m}'^4 \rightarrow \mathbf{m}_f\}$ (see Table 3.3) are obtained. The time spent through this PWL trajectory is 802 t.u. The controlled flows, $\mathbf{w}$, at each segment are given in Table 3.4.

Applying the closed loop control developed in Section 3.5 assuming $\Theta = 0.01$ which satisfies inequality (3.13) and zero noise effect, the total time to drive the system from $\mathbf{m}_0$ to $\mathbf{m}_f$ through the path is obtained as 800 t.u. Under closed loop control, the evolution of some markings of places; and the controlled flows of some transitions are shown in Figure 3.10 and Figure 3.11, respectively.

Table 3.4: Controlled flows for the case study

transition	(m_0, m^1)	(m^1, m^2)	(m^2, m^3)	(m^3, m^4)	(m^4, m_f)
S_M1_A	0.0100	0.0100	0.0100	0.0099	0.0100
E_M1_A	0.0100	0.0100	0.0100	0.0099	0.0100
S_M2_A	0.0100	0.0100	0.0100	0.0099	0.0100
E_M2_A	0.0100	0.0100	0.0100	0.0099	0.0100
S_M2_B	0.0100	0.0100	0.0100	0.0099	0.0100
E_M2_B	0.0100	0.0100	0.0100	0.0099	0.0100
S_M1_B	0.0100	0.0100	0.0100	0.0099	0.0100
E_M1_B	0.0100	0.0100	0.0100	0.0099	0.0100
S_M3	0.0100	0.0120	0.01495	0.0100	0.0100
E_M3	0.0140	0.0120	0.01985	0.0100	0.0100
Out	0.0000	0.0000	0.0000	0.0000	0.0000

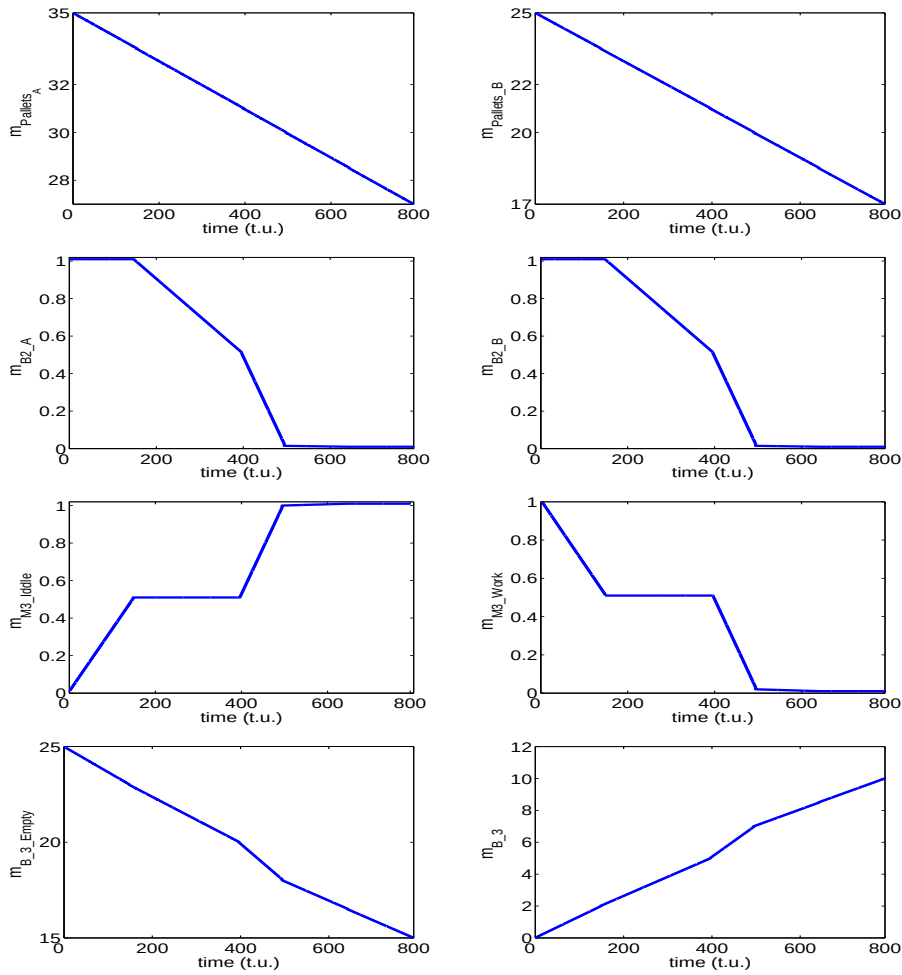


Figure 3.10: Evolution of some markings without noise effect for the case study

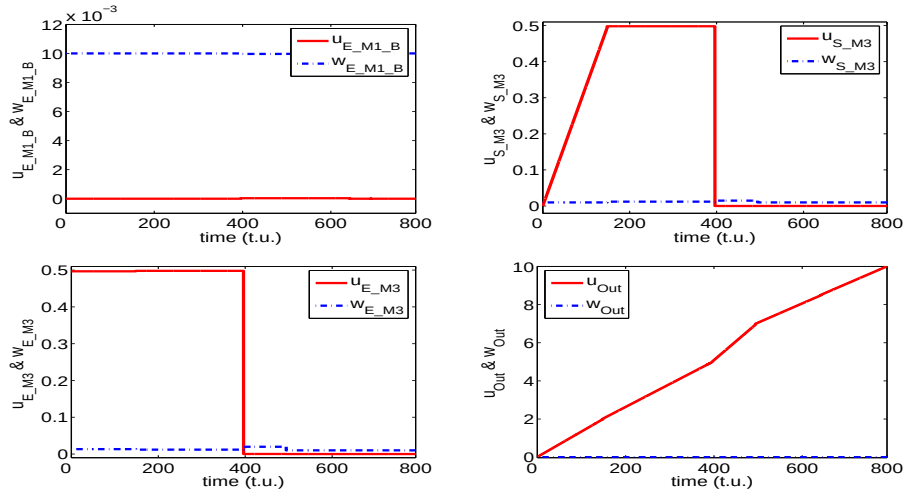


Figure 3.11: Evolution of control inputs and controlled flows of some transitions without noise effect for the case study

In order to see the noise effect on the closed loop controlled system, we introduce noise on the flow of transitions E_M1_B , S_M3 , E_M3 and Out during the time intervals $[0 \ 150]$ $[300 \ 400]$. The noise applied to transitions S_M3 , E_M3 , i.e. z_{S_M3} , z_{E_M3} (see equation (3.15)), are shown in Figure 3.12. Under this noisy condition, when the closed loop control is applied, the total time is obtained as 945 t.u. The controlled flows of some transitions and the evolution of markings of some places are shown in Figure 3.13 and Figure 3.14, respectively.

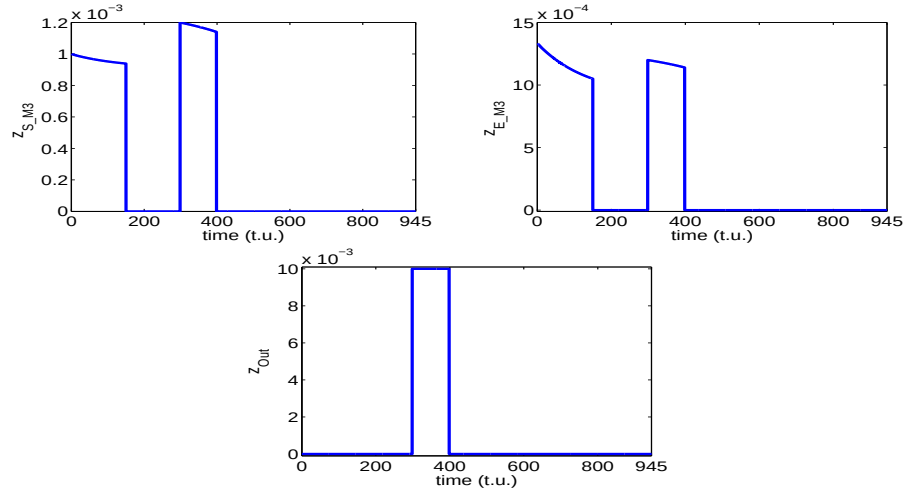


Figure 3.12: Noise effect on transitions for the case study

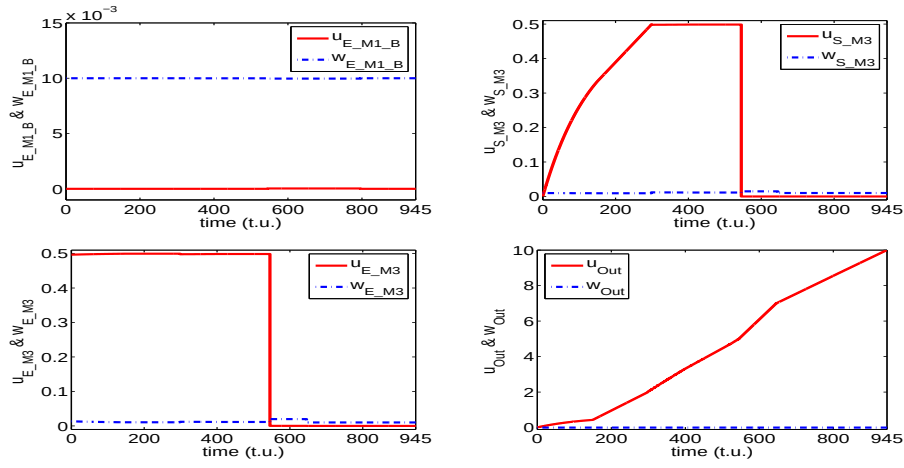


Figure 3.13: Evolution of control inputs and controlled flows of some transitions with noise effect for the case study

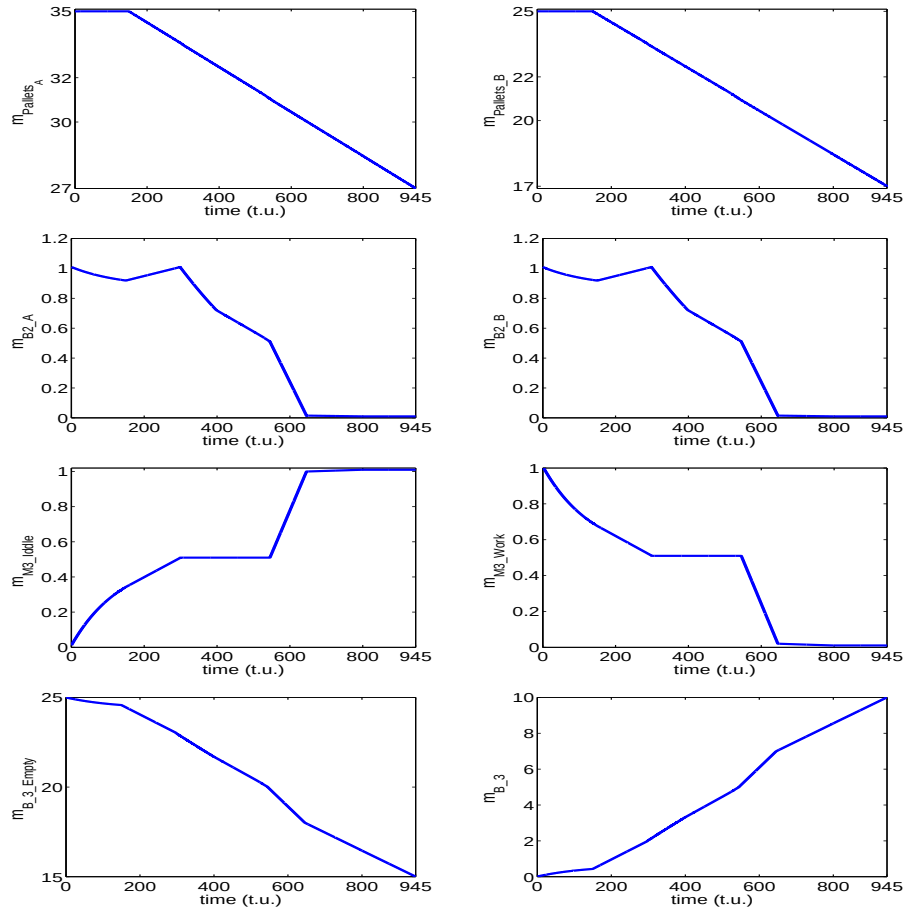


Figure 3.14: Evolution of some markings with noise effect for the case study

3.7 Summary

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

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

4. A FINAL MARKING PLANNING METHOD FOR JOIN FREE TIMED CONTINUOUS PETRI NETS

In this chapter, a control problem in which some components of the target marking are not specified is addressed. For calculation of unspecified components of the target marking under the objective of time minimization, the control strategy that we developed in Chapter 3 requests solving an NLP. In order to reduce computational complexity, we studied on a different control method. In order to start from a simple case, we limit our study to class of Join Free (JF) contPN. This method calculates the unspecified components of target markings by minimizing the time by means of a BLP and drives the system from initial marking to a fulfilled target marking by an online procedure.

4.1 Join Free contPNs

For JF contPN, at each marking the flow is limited by the same places (without synchronization). Hence, there exists only one constraint matrix $\mathbf{\Pi} \in \mathbb{R}_{\geq 0}^{|T| \times |P|}$ which is defined as:

$$\Pi_{ji} = \begin{cases} \frac{1}{Pre_{ij}}, & \text{if } Pre_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

The state equation of uncontrolled JF contPN is as follow:

$$\dot{\mathbf{m}} = \mathbf{C} \cdot \mathbf{\Lambda} \cdot \mathbf{\Pi} \cdot \mathbf{m} \quad (4.2)$$

The control action we will consider is same as the one we introduced in Section 3.2. According to this strategy, the overall behaviour of the system in which all transitions are controllable is ruled by the following system:

$$\begin{aligned} \dot{\mathbf{m}} &= \mathbf{C} \cdot [\mathbf{f} - \mathbf{u}] = \mathbf{C} \cdot \mathbf{w} \\ 0 &\leq \mathbf{u} \leq \mathbf{f} \end{aligned} \quad (4.3)$$

The constraint $0 \leq \mathbf{u} \leq \mathbf{f}$ can be rewritten as $0 \leq \mathbf{f} - \mathbf{u} \leq \mathbf{f}$. From the definition $\mathbf{w} = \mathbf{f} - \mathbf{u}$, the constraint can be expressed as:

$$0 \leq \mathbf{w} \leq \mathbf{\Lambda} \cdot \mathbf{\Pi} \cdot \mathbf{m} \quad (4.4)$$

4.2 Problem Statement

The main goal is to focus on the control problem for JF subclass in the case that some components of target markings are not specified. First, we develop an online control method and propose corresponding algorithm for driving JF contPN from its initial marking, $\mathbf{m}_0$, to target marking, $\mathbf{m}_f$, through a linear trajectory by minimizing the time. We propose a technique based on the developed control method to calculate the unspecified components of target merking under time minimization objective. Then the proposed algorithm is used to drive the system to fully specified target marking.

4.3 An Online Control Method for Join Free contPNs

In this section, an online control method that drives the system from the initial marking $\mathbf{m}_0$ to a desired target marking $\mathbf{m}_f$ through a linear trajectory will be introduced.

Our procedure consists of using discrete time representation of the system, and calculating control input at each sampling instant by using the maximum flows of transitions at $\mathbf{m}_0$ in the direction of $\mathbf{m}_f$ and maximum flows of transitions at $\mathbf{m}_f$ in the direction from $\mathbf{m}_0$ to $\mathbf{m}_f$.

Maximum flow of transitions at $\mathbf{m}_0$ in the direction of $\mathbf{m}_f$ is denoted by $\mathbf{w}_0$ and it is calculated by the following LPP, where $\mathbf{s}_0 = \mathbf{w}_0 \cdot \tau_0$:

$$\begin{aligned} \min_{\mathbf{s}_0} \quad & \tau_0 \\ \text{s.t.} \quad & \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s}_0 \quad (a) \\ & 0 \leq s_{0j} \leq \lambda_j \cdot \Pi_{ji} \cdot m_{0i} \cdot \tau_0 \\ & \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji} \neq 0 \quad (b) \end{aligned} \quad (4.5)$$

The equations correspond to: (a) the equation of the straight line connecting $\mathbf{m}_0$ to $\mathbf{m}_f$, (b) flow constraints at $\mathbf{m}_0$. Notice that (b) is a linear constraint because m_{0i} and m_{f_i} are known $\forall i \in \{1, 2, \dots, |P|\}$.

Maximum flow of transitions at $\mathbf{m}_f$ in the direction from $\mathbf{m}_0$ to $\mathbf{m}_f$ is

denoted by $\mathbf{w}_f$ and it is calculated by the following LPP, where $\mathbf{s}_f = \mathbf{w}_f \cdot \tau_f$:

$$\begin{aligned}
& \min_{\mathbf{s}_f} \quad \tau_f \\
& \text{s.t.} \quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s}_f \quad (a) \\
& \quad \quad 0 \leq s_{f_j} \leq \lambda_j \cdot \Pi_{ji} \cdot m_{f_i} \cdot \tau_f \\
& \quad \quad \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji} \neq 0 \quad (b)
\end{aligned} \tag{4.6}$$

The equations correspond to: (a) the equation of the straight line connecting $\mathbf{m}_0$ to $\mathbf{m}_f$, (b) flow constraints at $\mathbf{m}_f$. Notice that (b) is a linear constraint because m_{0i} and m_{fi} are known $\forall i \in \{1, 2, \dots, |P|\}$.

Proposition 4.1. *Let $\langle \mathcal{N}, \lambda, \mathbf{m}_0 \rangle$ be a contPN system with $\mathbf{m}_0 > \mathbf{0}$. If $\mathbf{m}_f$ belongs to $RS_{CL}^t(\mathcal{N}, \lambda, \mathbf{m}_0)$ and $\mathbf{m}_f > \mathbf{0}$, then LPPs in (4.5) and (4.6) are feasible.*

Proof. Since $\mathbf{m}_f$ is a reachable marking, then there exists $\mathbf{s}$ such that the state equations (4.5)(a) and (4.6)(a) are satisfied. By taking τ_0 and τ_f sufficiently large (4.5)(b) and (4.6)(b) can be satisfied since $\lambda_j \cdot \Pi_{ji}^z \cdot m_{0i} > 0$ and $\lambda_j \cdot \Pi_{ji}^z \cdot m_{fi} > 0$. $\square$

Since it is assumed that, $\mathbf{m}_0 > \mathbf{0}$ and $\mathbf{m}_f > \mathbf{0}$, then the linear trajectory from $\mathbf{m}_0$ to $\mathbf{m}_f$ can be followed by the system [49]. At each marking $\mathbf{m}$ on the line connecting $\mathbf{m}_0$ to $\mathbf{m}_f$ (i.e. $\mathbf{m} = \alpha \cdot \mathbf{m}_0 + (1 - \alpha) \cdot \mathbf{m}_f$, $\alpha \in [0, 1]$), the maximum flow in the direction to $\mathbf{m}_f$ is calculated by

$$\mathbf{w} = \alpha \cdot \mathbf{w}_0 + (1 - \alpha) \cdot \mathbf{w}_f \tag{4.7}$$

Corresponding control action is calculated easily by $\mathbf{u} = \mathbf{f} - \mathbf{w}$ where $\mathbf{f}$ and $\mathbf{w}$ are controlled and uncontrolled flow vectors, respectively.

In order to calculate control inputs to drive the system from $\mathbf{m}_0$ to $\mathbf{m}_f$ through a linear trajectory and drive the system to $\mathbf{m}_f$ by using the calculated control we propose to use discrete-time representation (3.12) of the contPN.

In order to calculate controlled flows and drive the system to $\mathbf{m}_f$, Algorithm 4.1 is developed. In this algorithm, controlled flow is calculated at each sampling instant by using the fact given in (4.7). Here, the controlled flows

$\mathbf{w}[0] = \mathbf{w}_0$ and $\mathbf{w}_f$ are obtained by solving LPPs in (4.5) and (4.6), respectively. Then, at the first iteration, $\mathbf{m}[1]$, α and $\mathbf{w}[1]$ (by using obtained α) are calculated. And obtained controlled flow is realized. At the next iteration, $\mathbf{m}[2]$, α and $\mathbf{w}[2]$ are calculated by similar way. This procedure is repeated until $\mathbf{m}_f$ is reached. During the execution of the algorithm, if α is obtained as bigger than 1, that is $\mathbf{m}_f$ is passed at the current iteration, θ is recalculated to reach $\mathbf{m}_f$ accurately.

Algorithm 4.1. Online Control

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

Solve LPP in (4.5) and LPP in (4.6)

$k=0$

while $\mathbf{m}[k] \neq \mathbf{m}_f$ **do**

Measure

$$\mathbf{m}[k+1] = \mathbf{m}[k] + \Theta \cdot \mathbf{C} \cdot \mathbf{w}[k] \quad (4.8)$$

Calculate corresponding α

$$\alpha = \frac{\mathbf{m}[k+1]_1 - \mathbf{m}_{f_1}}{\mathbf{m}_{0_1} - \mathbf{m}_{f_1}} \quad (4.9)$$

if $\alpha > 1$ **then**

$$\Theta = \frac{\mathbf{m}_{f_1} - \mathbf{m}[k]_1}{\mathbf{C}(1, \cdot) \cdot \mathbf{w}[k]} \quad (4.10)$$

end if

Advance one step

$$\mathbf{m}[k+1] = \mathbf{m}[k] + \Theta \cdot \mathbf{C} \cdot \mathbf{w}[k] \quad (4.11)$$

Calculate corresponding $\mathbf{w}[k+1]$

$$\mathbf{w}[k+1] = \alpha \cdot \mathbf{w}_0 + (1 - \alpha) \cdot \mathbf{w}_f \quad (4.12)$$

$k = k + 1$

end while

Example 4.1. Let us consider JF contPN in Figure 4.1 with $\lambda = [1 \ 1 \ 1 \ 1 \ 1 \ 1]^T$. The only minimal P-semiflow is $\mathbf{y} = [2 \ 2 \ 1 \ 1 \ 1]$ and there are two minimal T-semiflows $\mathbf{x}^1 = [1 \ 1 \ 0 \ 0 \ 0 \ 0]^T$ and $\mathbf{x}^2 = [0 \ 0 \ 1 \ 1 \ 1 \ 1]^T$. Our aim is to drive the system from $\mathbf{m}_0 = [13 \ 3 \ 4 \ 4 \ 5]^T$ to a final state $\mathbf{m}_f = [10 \ 6 \ 6 \ 3 \ 2]^T$ by using the proposed control method. Let us assume that $\theta = 0.01$.

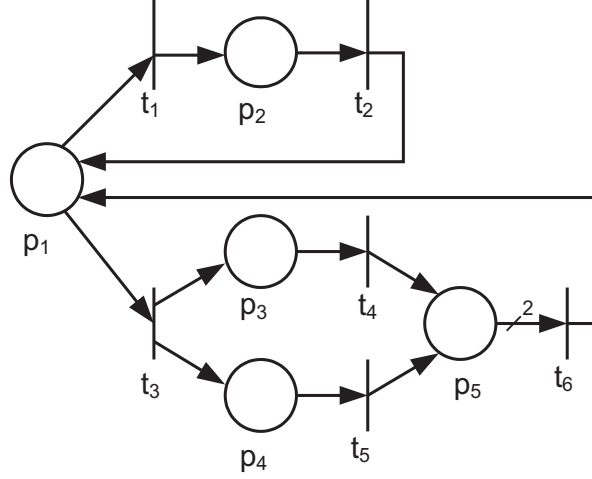


Figure 4.1: A JF Petri net taken from [4]

The system dynamics can be described as follows:

$$\begin{aligned}
 \dot{m}_1 &= f_2 + f_6 - f_1 - f_3 = \lambda_2 \cdot m_2 + \frac{\lambda_6}{2} \cdot m_5 - \lambda_1 \cdot m_1 - \lambda_3 \cdot m_1 \\
 \dot{m}_2 &= f_1 - f_2 = \lambda_1 \cdot m_1 - \lambda_2 \cdot m_2 \\
 \dot{m}_3 &= f_3 - f_4 = \lambda_3 \cdot m_1 - \lambda_4 \cdot m_3 \\
 \dot{m}_4 &= f_3 - f_5 = \lambda_3 \cdot m_1 - \lambda_5 \cdot m_4 \\
 \dot{m}_5 &= f_4 + f_5 - 2 \cdot f_6 = \lambda_4 \cdot m_3 + \lambda_5 \cdot m_4 - \lambda_6 \cdot m_5
 \end{aligned} \tag{4.13}$$

At $\mathbf{m}_0$, solving LPP (4.5) yields $\mathbf{w}_0 = [5.8 \ 2.05 \ 2.5 \ 0 \ 1.25 \ 2.5]^T$ $\tau_0 = 0.8$ t.u. At $\mathbf{m}_f$, solving LPP (4.6) yields $\mathbf{w}_f = [4.04 \ 2.54 \ 1 \ 0 \ 0.5 \ 1]^T$ $\tau_f = 2$ t.u.

By executing Algorithm 4.1, $\mathbf{m}_f$ after 122 discrete steps, which corresponds to 1.22 t.u. The evolution of m_1 and m_2 and the controlled flows of transitions t_1 and t_2 are shown in Figures 4.2 and 4.3, respectively.

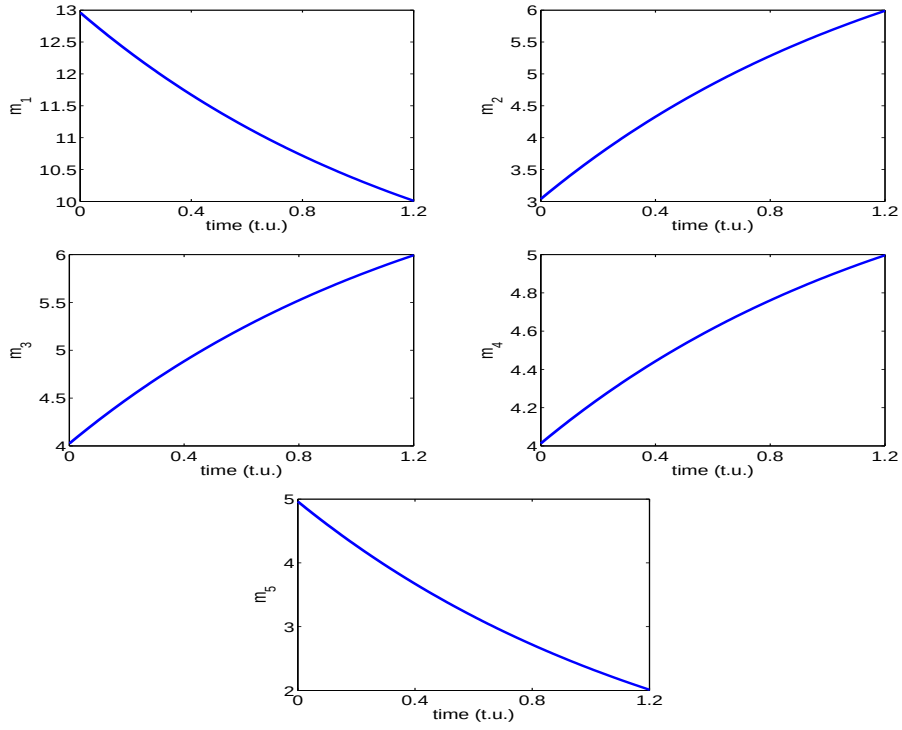


Figure 4.2: Evolution of markings for Example 4.1

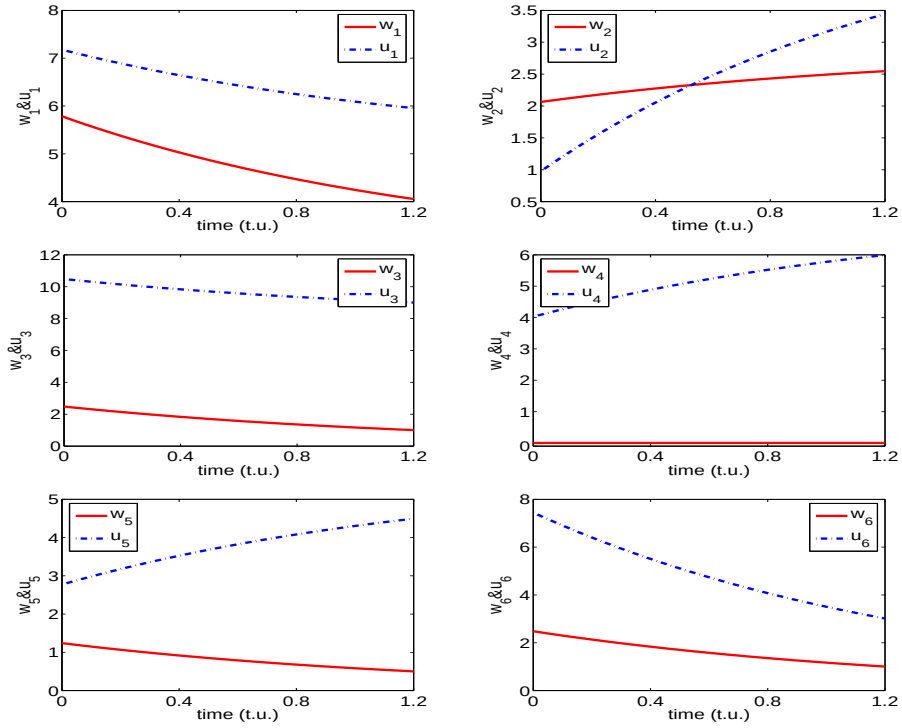


Figure 4.3: Evolution of control action, u , and controlled flow, w , for Example 4.1

4.4 Final Marking Planning for Join Free contPNs

So far, we introduced an online control method for driving the system from a given initial state $\mathbf{m}_0$ to the specified target marking. In this section, we consider the control problems that final markings of some places are specified, while the others are not. The set of places whose final markings are specified precisely is denoted by $P_{sp} = \{p_i \mid m_{f_i} \text{ is specified}\}$ and the set of places whose final markings are not specified is denoted by $P_{un} = \{p_i \mid m_{f_i} \text{ is un-specified}\}$. In order to calculate the $m_{f_i} \forall p_i \in P_{un}$ by minimizing the time (or maximizing controlled flows $\mathbf{w}_0$ and $\mathbf{w}_f$), Algorithm 4.1 is executed again with only one modification. Thus, as differ from the first case we propose to solve BLP in (4.14) at which $\mathbf{s}_0 = \mathbf{w}_0 \cdot \tau_0$, $\mathbf{s}_f = \mathbf{w}_f \cdot \tau_f$:

$$\begin{aligned}
 &\text{Variables: } \tau_0, \tau_f, \mathbf{s}_0, \mathbf{s}_f, m_{f_i} \forall p_i \in P_{un} \\
 &\min \quad \tau_0 + \tau_f \\
 &\text{s.t.} \quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s}_0 \quad (a1) \\
 &\quad 0 \leq s_{0j} \leq \lambda_j \cdot \Pi_{ji} \cdot m_{0i} \cdot \tau_0 \\
 &\quad \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji} \neq 0 \quad (a2) \\
 &\quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \mathbf{s}_f \quad (b1) \\
 &\quad 0 \leq s_{fj} \leq \lambda_j \cdot \Pi_{ji} \cdot m_{f_i} \cdot \tau_f \\
 &\quad \forall j \in \{1, \dots, |T|\} \text{ where } i \text{ satisfies } \Pi_{ji} \neq 0 \quad (b2) \\
 &\quad \mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma}, \boldsymbol{\sigma} > \mathbf{0} \quad (c)
 \end{aligned} \tag{4.14}$$

The equations correspond to: (a1)&(b1) the time dependent equation of the straight line connecting $\mathbf{m}_0$ to $\mathbf{m}_f$; (a2)&(b2) flow constraints at $\mathbf{m}_0$ and $\mathbf{m}_f$, respectively; (c) reachability condition of $\mathbf{m}_f$. Note that, if the net is consistent and every transition can fire at least once (4.14)(c) is equivalent to $\mathbf{B}_y^T \cdot \mathbf{m}_f = \mathbf{B}_y^T \cdot \mathbf{m}_0$, $\mathbf{m}_f > \mathbf{0}$ (4.14)(c') (see Proposition 2.1).

Example 4.2. Let us consider JF contPN in Figure 4.1 again with the same initial marking $\mathbf{m}_0 = [13 \ 3 \ 4 \ 4 \ 5]^T$. Final marking of some places (not all) are specified: $m_{f_1} = 10$, $m_{f_4} = 3$, $m_{f_5} = 2$, that is $P_{sp} = \{p_1, p_4, p_5\}$ and

$P_{un} = \{p_2, p_3\}$. Our objective is to find the final markings of m_{f_2} and m_{f_3} by minimizing the time and drive the system to fulfilled final marking.

By solving the BLP in (4.14), markings of unspecified places are obtained as $m_{f_2} = 6.5$ and $m_{f_3} = 5$, that is $\mathbf{m}_f = [10 \ 6.5 \ 5 \ 3 \ 2]^T$. By Algorithm 4.1, $\mathbf{m}_f$ is reached by 90 discrete steps (0.90 t.u.). The evolution of markings and the controlled flows of transitions are shown in Figures 4.4 and 4.5, respectively.

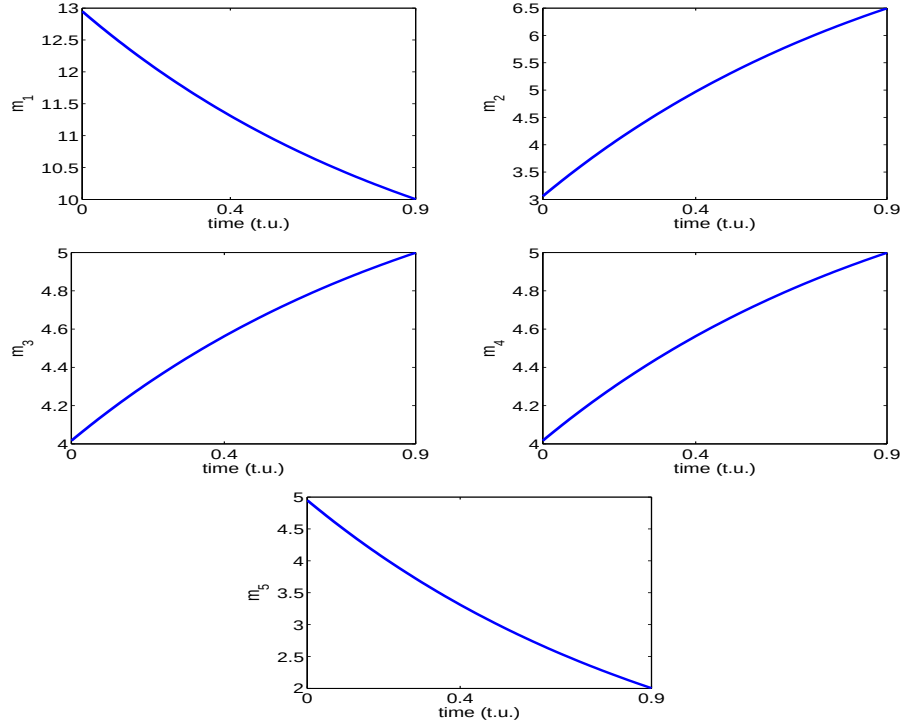


Figure 4.4: Evolution of markings for Example 4.2

4.5 Summary

In this chapter, an online control method is developed for JF contPN. The method takes discrete time representation. In this method, in order to drive the system from its initial marking to target marking, corresponding control action is calculated and applied at each time step. Algorithm 4.1 which uses LPP is developed for this method.

In some control problems, target markings of some places are given while that of others are not specified. In that case, we propose to calculate

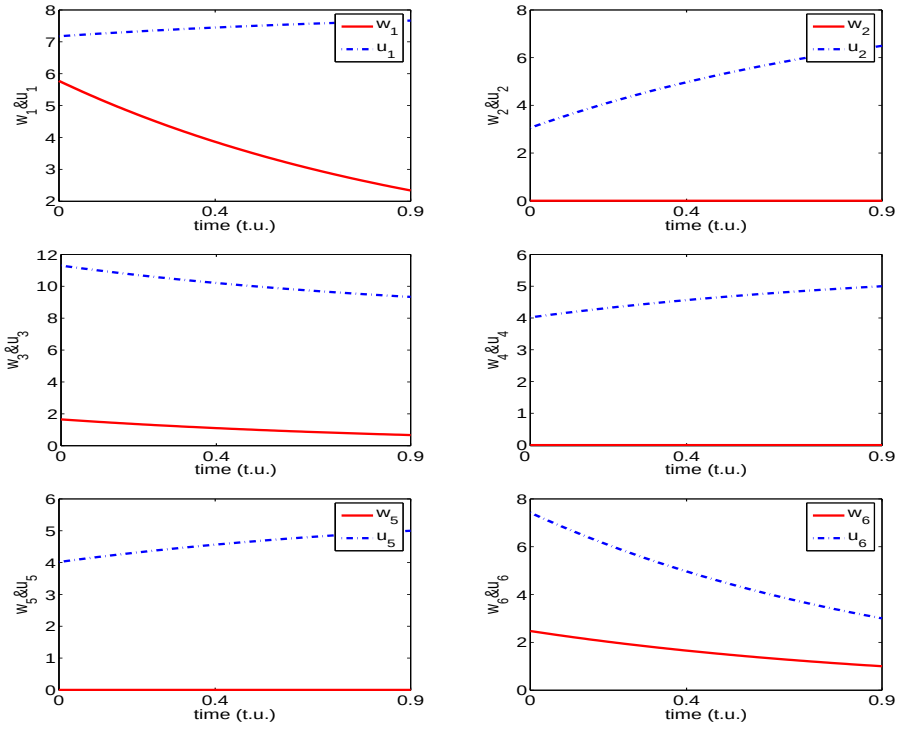


Figure 4.5: Evolution of $\mathbf{u}$, $\mathbf{w}$ for Example 4.2

unspecified target markings of places by solving a BLP with time minimization objective. Then Algorithm 4.1 is executed again with a simple modification.

5. A CONTROL METHOD FOR DISTRIBUTED TIMED CONTINUOUS PETRI NETS

In this chapter, we give a definition for distributed timed continuous Petri nets with infinite server semantics (DcontPN). First, DcontPN composed by two subsystems is considered and a distributed control algorithm is developed. Then, it is shown that the algorithm proposed for the case of distributed systems composed only by two subsystems can be easily extended in order to be able to deal with distributed systems composed by a finite number of subsystems. Finally, a manufacturing system containing assembly operations is taken as a case study.

5.1 Distributed Timed Continuous Petri nets

This section introduces the main concepts related to distributed timed continuous Petri net and presents an introductory examples.

Definition 5.1 (DcontPN). *A Distributed timed continuous Petri net (DcontPN) system is a set of timed continuous Petri net systems (called as subsystems) interconnected through channels/buffers modeled as places (called as channel places).*

A DcontPN system is composed by several timed continuous Petri net subsystems that could model different parts of a plant, interconnected by channels (or buffers) modeled by places. These channels have finite capacity and contain the products produced by a given subsystem and needed by another subsystem.

Example 5.1. *Let us consider a car manufacturing factory composed by two plants A and D in two different cities as a simple example. The Petri net model is given in Figure 5.1. The plant A produces the car body (place p_1) and then sends it to the plant D (place p_2). The plant A can produce a limited number of car bodies in parallel (the initial marking of p_3). In plant D, the engine is constructed (p_4) and then it is put in an intermediate buffer p_5 . The same plant*

paints the body received from plant A in p_6 and puts it in p_7 to be assembled together with the engine. The firing of t_8 means the production of a new car. We assume that plant D can produce a limited number of engines in parallel (initial marking of p_8) and can paint a limited car bodies in parallel (initial marking of p_9). Place p_a is the buffer containing the car bodies produced by plant A while p_b is the buffer containing the finished products. Since we don't want to produce more than we sell, the plant A starts to produce a new body (firing of t_1) only when a car is sold.

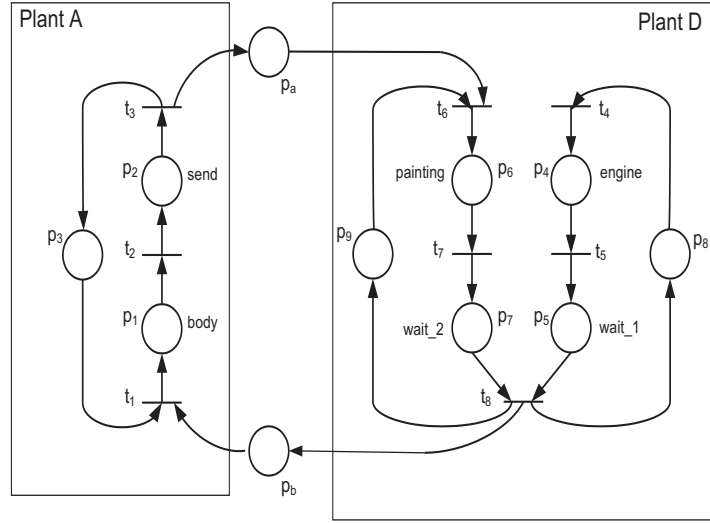


Figure 5.1: A DcontPN modeling a car manufacturing plant where p_a and p_b are channel places.

Let K denote the set of subsystems of a given DcontPN. The set of places, transitions and token flow matrix of subsystem $k \in K$ is denoted by P^k , T^k and $C^k \in \mathbb{R}^{|P^k| \times |T^k|}$, respectively. We assume, $P^k \cap P^l = \emptyset$ and $T^k \cap T^l = \emptyset$, $\forall k, l \in K$, $k \neq l$. The directional connection between subsystems is provided by a set of places called *channel* or *buffer places*. In particular, the directional connection from subsystem k to l is provided by a set of places denoted $B^{(k,l)}$, whose input transitions are contained only in subsystem k and output transitions are contained only in subsystem l , i.e., $B^{(k,l)} = \{p \in P \mid p \bullet \in T^k, p \bullet \in T^l, p \notin P^q \ \forall q \in K\}$ for every $k, l \in K$, $k \neq l$, and $B^{(l,l)} = \emptyset$ for every $l \in K$.

Note that $p \in B^{(k,l)}$ is an *input buffer* of subsystem l and an *output buffer* of subsystem k . The set of all output buffers of subsystem k is denoted by $B^{(k,*)}$, i.e., $B^{(k,*)} = \bigcup_{l \in K} B^{(k,l)}$, and the set of all input channels of subsystem k is denoted by $B^{(*,k)}$, i.e., $B^{(*,k)} = \bigcup_{l \in K} B^{(l,k)}$.

The marking vector of a subsystem k is denoted by $\mathbf{m}(P^k) \in \mathbb{R}_{\geq 0}^{|P^k|}$. When designing a controller, it must be taken into account that the controller of a given subsystem only knows its marking and the marking of its input buffers, i.e., the marking of the other subsystems and their input buffers are not observable.

Example 5.2. *Let us consider the DcontPN given in Figure 5.1. It is composed of two subsystems. For the first subsystem (Plant A), the set of places and transitions are: $P^1 = \{p_1, p_2, p_3\}$, $T^1 = \{t_1, t_2, t_3\}$; while they are $P^2 = \{p_4, p_5, p_6, p_7, p_8, p_9\}$ and $T^2 = \{t_4, t_5, t_6, t_7, t_8\}$ for the second one (plant D). These two subsystems communicate through two channels: p_a for the communication from subsystem 1 to subsystem 2 and p_b for the communication from subsystem 2 to subsystem 1. Hence, $B^{(1,2)} = \{p_a\}$ and $B^{(2,1)} = \{p_b\}$, implying $B^{(*,1)} = B^{(2,*)} = \{p_b\}$ and $B^{(*,2)} = B^{(1,*)} = \{p_a\}$. Finally, the token flow matrices of subsystems are:*

$$C^1 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \quad C^2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

5.2 Problem Statement

The main goal is to drive subsystems of a DcontPN from their initial marking to their target marking. First, the problem of reaching a target marking in a DcontPN composed by two subsystems is considered, then it is extended to more general structures. The controllers should be computed offline, and

it is assumed that the structure and state of each subsystem is unknown to the controller located at the other subsystem. In order not to overload the communication network, it is desirable to interchange as few information as possible between controllers. The underlying idea of the strategy is to design a local controller for each subsystem in contrast to the centralized control. Each controller computes a control law to reach the target state of its subsystem, and asks other controllers to produce enough resources in the buffers to execute its control actions. The main focus is reaching the target markings of both subsystems in parallel after a finite amount of time.

Given that the subsystems are interconnected, they may require resources to be available in the communication buffers to reach the target marking. The following example illustrates this situation.

Example 5.3. *Consider the DcontPN in Figure 5.1 with $\mathbf{m}_0(P^1) = [0 \ 1 \ 3]^T$, $\mathbf{m}_0(P^2) = [0 \ 0 \ 0 \ 0 \ 2 \ 2]^T$, $m_0(p_a) = 0$, $m_0(p_b) = 1$ and let $\mathbf{m}_f(P^1) = [0 \ 1 \ 3]^T$, $\mathbf{m}_f(P^2) = [0 \ 0 \ 1 \ 0 \ 2 \ 1]^T$ be the target markings of each subsystem. Let the flow integrals of subsystem 1 and 2 be denoted as $\mathbf{s}^1$ and $\mathbf{s}^2$ respectively.*

Let us consider that each subsystem is isolated from the rest of the system. A controller for the second subsystem could compute $s^2(t_6) = 1$, $s^2(t_4) = s^2(t_5) = s^2(t_7) = s^2(t_8) = 0$ so that the subsystem reaches the target marking. Given that the initial marking and target marking of subsystem 1 are the same, a controller for that subsystem could yield: $s^1(t_1) = s^1(t_2) = s^1(t_3) = 0$. Since $m_0(p_a) = 0$, transition t_6 cannot fire unless t_3 fires. Unfortunately, according to the computed controls, t_3 will not fire ($s^1(t_3) = 0$). Hence, these controls are not valid to reach the desired target marking of subsystem 2. In order to solve this situation, subsystem 2 may ask subsystem 1 to put enough tokens in p_a . This can be achieved easily by firing t_3 . However, this will imply that subsystem 1 moves away from its desired target marking.

Apart from the problem of tokens (resources) required in the buffer places at the initial time, it could happen that the target markings cannot be reached due to the system structure and initial marking. The following example deals with this case.

Example 5.4. Consider again the DcontPN in Figure 5.1. For subsystem 1, let the target marking be $\mathbf{m}_f(P^1) = [0 \ 0 \ 3]^T$ which is reachable from $\mathbf{m}_0(P^1) = [0 \ 0 \ 3]^T$ locally. For subsystem 2, let the target marking be $\mathbf{m}_f(P^2) = [0 \ 0 \ 1 \ 0 \ 2 \ 1]^T$ which is reachable from $\mathbf{m}_0(P^2) = [0 \ 0 \ 0 \ 0 \ 2 \ 2]^T$ locally by firing t_6 , i.e., if it is considered isolated from the rest of the system. But when both subsystems are connected through the buffers p_a and p_b with $m_0(p_a) = m_0(p_b) = 0$, the target markings are locally reachable but not globally reachable.

When stating the problem we are implicitly assuming that all target markings of subsystems are reachable, meaning that the final marking of the overall net system, i.e., the net containing all subsystems and buffers, is reachable.

5.3 A controller for DcontPN

This section is devoted to the design of a distributed controller for a particular class of DcontPN: a DcontPN composed by only two subsystems interconnected. We will first present the algorithm associated to the local controller of each subsystem and then prove its correctness.

The control algorithm that will be presented applies to those DcontPN systems satisfying the following assumptions:

- (A1) The target marking $\mathbf{m}_f$ is strictly positive and reachable at the overall system.
- (A2) The DcontPN is composed of two MTS subsystems. The minimal T-semiflows are denoted by $\mathbf{x}^1$ and $\mathbf{x}^2$.
- (A3) The overall system is MTS.

The first assumption is simply a necessary condition for reachability of the target markings. The second assumption reduces the class of DcontPN to those systems having two subsystems that are MTS while the third one states that the overall system is MTS.

In order to drive the subsystems from their initial states to the target states, Algorithm 5.1 is developed. It has 7 basic steps and represents the local controller that will be executed in each subsystem separately. We assume that there exist channels between controllers that allow the interchange of information. These information are used to obtain an agreement between controllers ensuring that the final marking is globally reached.

During the initialization, the algorithm requires the initial and target markings, the token flow matrix, the input and output buffers and the initial marking of the input buffers of the corresponding subsystem. In step 1, each subsystem computes the flow integral $\mathbf{z}$ required to reach its target marking without taking into account the marking of the buffers, this step is carried out by means of the LPP (5.1). Step 2 computes the amounts of tokens q_p^{req} to be produced in each input buffer p in order to be able to fire $\mathbf{z}$. The connected subsystems are informed about the amounts of required tokens r_p^{req} in step 3. In step 4, each subsystem receives the amount of tokens it has to produce (if any) in its output buffers.

In step 5, it is computed how many tokens would remain in each output buffer if the present control was applied. If this value is negative, more tokens must be produced in the output buffers, and therefore the control law must be recomputed. This re-computation is achieved in step 6 using LPP (5.2). Observe that comparing with LPP (5.1) of step 1 only one extra constraint is added in order to ensure that enough tokens are produced in the output buffers.

Example 5.5. *Let us go back to the DcontPN in Figure 5.1 to illustrate the process of Algorithm 5.1. Assume: $\mathbf{m}_0(P^1) = [0 \ 0 \ 3]^T$, $\mathbf{m}_0(P^2) = [0 \ 0 \ 0 \ 0 \ 2 \ 2]^T$, $m_0(p_a) = 0$, $m_0(p_b) = 1$, $\mathbf{m}_f(P^1) = [0 \ 0 \ 3]^T$, $\mathbf{m}_f(P^2) = [0 \ 1 \ 0 \ 1 \ 1 \ 1]^T$. The steps of Algorithm 5.1 applied to subsystem 1 and subsystem 2 are shown in Table 5.5. Observe that subsystem 1 needs to recompute its initial control law. This is because its initial and final marking are the same, thus the initial control is $\mathbf{z} = \mathbf{0}$ while the request of subsystem 2 is to produce one token in p_a .*

Algorithm 5.1. Distributed controller of subsystem $k \in \{1, 2\}$

Input: $C^k, m_0(P^k), m_f(P^k), B^{(k,*)}, B^{(*,k)}$

Output: flow integral vector s

1) Solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \mathbf{z} \\ \text{s.t.} \quad & \mathbf{m}_f(P^k) - \mathbf{m}_0(P^k) = \mathbf{C}^k \cdot \mathbf{z}, \\ & \mathbf{z} \geq 0 \end{aligned} \tag{5.1}$$

2) For every $p \in B^{(*,k)}$ calculate

$$q_p^{req} = \left(\sum_{t \in p^\bullet} Pre(p, t) \cdot z(t) \right) - m_0(p)$$

3) For all $p \in B^{(*,k)}$ send q_p^{req} to the connected subsystem

4) For all $p \in B^{(k,*)}$ receive r_p^{req} from the connected subsystem

5) For all $p \in B^{(k,*)}$ calculate

$$h_p = \left(\sum_{t \in \bullet p} Post(p, t) \cdot z(t) \right) - r_p^{req}$$

6) **If** $\min_{p \in B^{(k,*)}} \{h_p\} < 0$ **then** solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \mathbf{s} \\ \text{s.t.} \quad & \mathbf{m}_f(P^k) - \mathbf{m}_0(P^k) = \mathbf{C}^k \cdot \mathbf{s}, \\ & \left(\sum_{t \in \bullet p} Post(p, t) \cdot s(t) \right) \geq r_p^{req}, \forall p \in B^{(k,*)} \\ & \mathbf{s} \geq 0 \end{aligned} \tag{5.2}$$

Else

$$\mathbf{s} = \mathbf{z}$$

End_If

7) **return** s

The re-computation is done in step 6 and the solution is simply firing once its T -semiflow.

Table 5.1: Execution of the algorithms on the DcontPN in Figure 5.1

	Subsystem 1	Subsystem 2
Step 1)	$z(t_1) = z(t_2) = z(t_3) = 0$	$z(t_4) = z(t_5) = 1$ $z(t_6) = z(t_7) = 1$ $z(t_8) = 0$
Step 2)	$q_{p_b}^{REQ}(\mathbf{z}) = z(t_1) - m_0(p_b)$ $= -1$	$q_{p_a}^{REQ}(\mathbf{z}) = z(t_6) - m_0(p_a)$ $= 1$
Step 3)	Send $q_{p_b}^{REQ} = -1$	Send $q_{p_a}^{REQ} = 1$
Step 4)	Receive $r_{p_a}^{REQ} = 1$	Receive $r_{p_b}^{REQ} = -1$
Step 5)	$h_{p_a}(\mathbf{z}) = z(t_3) - r_{p_a}^{REQ}$ $= -1$	$h_{p_b}(\mathbf{z}) = z(t_8) - r_{p_b}^{REQ}$ $= 1$
Step 6)	Since $h_{p_a} < 0$, LPP (5.2) has to be solved, resulting: $s(t_1) = s(t_2) = s(t_3) = 1$	Since $h_{p_b} \geq 0$, $\mathbf{s} = \mathbf{z}$
Step 7)	return $\mathbf{s} = [1 \ 1 \ 1]^T$	return $\mathbf{s} = \mathbf{z} = [1 \ 1 \ 1 \ 1 \ 0]^T$

The necessity of assumptions (A2) or (A3) is illustrated in the following examples.

Example 5.6. *Let us consider the net in Figure 5.2. Both subsystem 1 and subsystem 2 are MTS. But the overall system is not conservative. For this DcontPN $\mathbf{m}_f(P^1) = [0 \ 1]^T$, $\mathbf{m}_f(P^2) = [1 \ 0]^T$ are reachable from initial conditions $\mathbf{m}_0(P^1) = [1 \ 0]^T$, $\mathbf{m}_0(P^2) = [0 \ 1]^T$ while $\mathbf{m}_0(p_a) = 0$ $\mathbf{m}_0(p_b) = 0$.*

By executing Algorithm 5.1, at the side of subsystem 1 $z(t_1) = 1$, $z(t_2) = 0$ values are obtained, while $z(t_3) = 0$, $z(t_4) = 1$ are obtained at the side of subsystem 2. Observe that, this control law yields negative channels, that which $m(p_a) = -1$ and $m(p_b) = -1$. Hence, at the side of both subsystems, the flow integrals are recomputed at step 6 and the flow integrals of transitions are obtained as $s(t_1) = 1.5$, $s(t_2) = 0.5$, $s(t_3) = 1$, $s(t_4) = 2$. Unfortunately, even these flow integrals yield negative channels, i.e. $m(p_a) = -0.5$ $m(p_b) = -1$.

In order to obtain proper control law, even re-iterations (repeating the procedure from step 2 to step 7) does not make channels nonnegative. This is illustrated in Table 5.2.

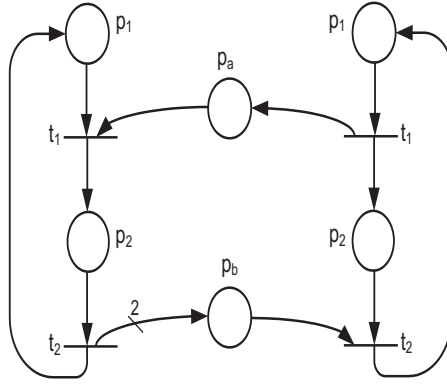


Figure 5.2: A simple DcontPN which is non-MTS

Example 5.7. *Let us consider the net in Figure 5.3. Subsystem 1 is MTS, but subsystem 2 is non-MTS since it has two T-semiflows: $[1 \ 0 \ 1 \ 0 \ 1]^T$ and $[0 \ 1 \ 0 \ 1 \ 1]^T$. For this DcontPN $\mathbf{m}_f(P^1) = [4 \ 2 \ 1]^T$, $\mathbf{m}_f(P^2) = [2 \ 2 \ 1 \ 1]^T$ are*

reachable from initial states $\mathbf{m}_0(P^1) = [1 \ 3 \ 3]^T$, $\mathbf{m}_0(P^2) = [1 \ 1 \ 2 \ 2]^T$ while $\mathbf{m}_0(p_a) = 0$ $\mathbf{m}_0(p_b) = 0$.

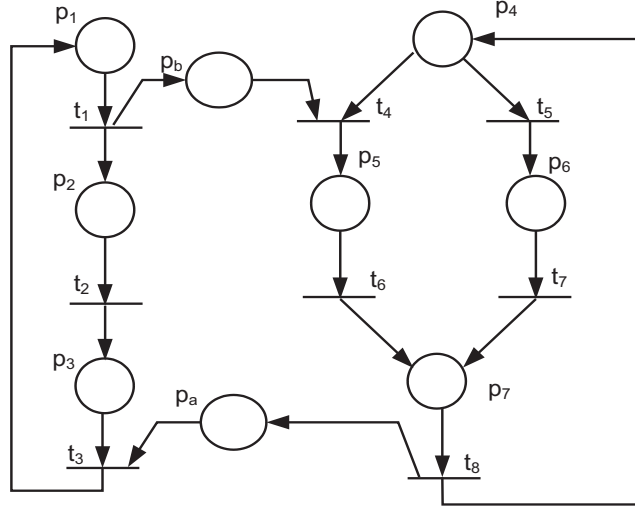


Figure 5.3: A DcontPN

Since the subsystem 2 has two T -semiflows LPPs in Algorithm 5.1 have more than one (two) possible solutions. At step 1 of Algorithm 5.1, $z(t_1) = 0$, $z(t_2) = 1$, $z(t_3) = 3$ and at the side of subsystem 2 $z(t_4) = 1$, $z(t_5) = 0$, $z(t_6) = 0$, $z(t_7) = 1$, $z(t_8) = 2$ values are obtained. This values do not yield nonnegative output channels, therefore control law of each subsystem has to be recalculated at step 6. For subsystem 1, a unique solution exists such that $s(t_1) = 1$, $s(t_2) = 2$, $s(t_3) = 4$.

For subsystem 2, two different solutions can be obtained at step 6: solution 2₁: $s(t_4) = 2$, $s(t_5) = 0$, $s(t_6) = 1$, $s(t_7) = 1$, $s(t_8) = 3$ which yields $m(p_a) = -1$ $m(p_b) = -1$. solution 2₂: $s(t_4) = 1$, $s(t_5) = 1$, $s(t_6) = 0$, $s(t_7) = 2$, $s(t_8) = 3$ which yields $m(p_a) = -1$ $m(p_b) = 0$. Hence, non of these solutions result in an appropriate flow integrals for driving subsystems to their target marking by keeping the channels nonnegative.

Let us try to re-iterate as in Example 5.6. If we start re-iteration by using solution 2₁, we see that even infinite number of iteration does not yield appropriate control law to drive subsystems from their initial marking to target marking by keeping the channels nonnegative.

Table 5.2: Execution of the Algorithm 5.2 on the DcontPN in Example 5.6

	Subsystem 1	Subsystem 2	$m(p_a)$	$m(p_b)$
Iteration 1	$s(t_1) = 1 \quad s(t_2) = 0$	$s(t_3) = 0 \quad s(t_4) = 1$	-1	-1
Iteration 2	$s(t_1) = 1.5 \quad s(t_2) = 0.5$	$s(t_3) = 1 \quad s(t_4) = 2$	-0.5	-1
Iteration 3	$s(t_1) = 2.5 \quad s(t_2) = 1.5$	$s(t_3) = 2.5 \quad s(t_4) = 3.5$	0	-0.5
Iteration 4	$s(t_1) = 2.75 \quad s(t_2) = 1.75$	$s(t_3) = 2.5 \quad s(t_4) = 3.5$	-0.25	0
Iteration 5	$s(t_1) = 2.75 \quad s(t_2) = 1.75$	$s(t_3) = 2.75 \quad s(t_4) = 3.75$	0	-0.25
Iteration 6	$s(t_1) = 2.875 \quad s(t_2) = 1.875$	$s(t_3) = 2.75 \quad s(t_4) = 3.75$	-0.125	0
Iteration 7	$s(t_1) = 2.875 \quad s(t_2) = 1.875$	$s(t_3) = 2.875 \quad s(t_4) = 3.875$	0	-0.125
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Let us show that Algorithm 5.1 computes a control law for both subsystems that ensures the reachability of their target markings. Moreover, there exists a (finite) time instant at which both subsystems are in their target markings.

Theorem 5.1. *Let $\mathcal{N}$ be a DcontPN satisfying assumptions (A1), (A2) and (A3), and let $\mathbf{s}^1$ and $\mathbf{s}^2$ be the flow integral vectors computed by Algorithm 5.1 for each subsystem for a given initial and target markings. The application of $\mathbf{s}^1$ and $\mathbf{s}^2$ drives the subsystems to their target markings.*

Proof. At step 2 of Algorithm 5.1, each controller computes a flow integral vector, $\mathbf{z}^1$ and $\mathbf{z}^2$ for subsystem 1 and 2 respectively, without considering the

marking of the input buffers. If the initial marking of every buffer p is high enough, h_p is nonnegative and the flow integrals $\mathbf{z}^1$ and $\mathbf{z}^2$ can be applied. After a technical remark, it will be shown that at most one subsystem can have output buffers with negative h_p . Afterwards, it is shown that if such a subsystem exists, step 7 computes an appropriate integral vector to reach the target marking while keeping the buffers at nonnegative markings.

Let $\mathbf{m}_0(P^1)$ and $\mathbf{m}_0(P^2)$ be the initial markings of the subsystems, and $\mathbf{m}_f(P^1)$ and $\mathbf{m}_f(P^2)$ be the target markings. By assumption (A2), subsystem 1 and subsystem 2 are MTS. Therefore, the only flow integral vectors of the subsystems to reach the target markings are obtained by adding (or subtracting) instances of the T-semiflows to an integral vector that is solution of the fundamental equation. Hence, all the possible integral vectors are defined by $\mathbf{z}^1 + k^1 \cdot \mathbf{x}^1$ and $\mathbf{z}^2 + k^2 \cdot \mathbf{x}^2$ respectively, where $\mathbf{z}^1$ and $\mathbf{z}^2$ are the solutions yielded at step 2, $\mathbf{x}^1$ and $\mathbf{x}^2$ are the T-semiflows of the subsystems, and k^1 and k^2 are real numbers.

Given that the whole system is MTS, recall assumption (A3), for every $p \in B^{(1,2)}$ and every $q \in B^{(2,1)}$ there exists a P-semiflow that contains p and q . Therefore, k^1 and k^2 cannot be chosen independently to reach the target makings: if k^1 is increased (decreased), the marking of the output buffers of subsystem 1 will increase(decrease) and the marking of its input buffers will decrease (increase). A similar phenomenon occurs if k_2 is increased(decreased). Obviously the values of k_1 and k_2 must be such that the markings of buffers remain nonnegative.

Assume that after step 5, there exist $p \in B^{(1,2)}$ and $q \in B^{(2,1)}$ such that $h_p < 0$ and $h_q < 0$. According to the above remark, any positive or negative k^1 and k^2 would decrease the marking of at least one buffer. That is, it is not possible to reach the target marking and have nonnegative markings at buffers. Hence, the target marking is not reachable at the overall system what contradicts assumption (A1).

Let us assume without loss of generality that there exists $p \in B^{(1,2)}$ such that $h_p < 0$, and that for every $q \in B^{(2,1)}$ it holds $h_q > 0$. Let $\mathbf{s}^1$ be the solution

obtained for subsystem 1 at step 6 and $\mathbf{s}^2$ be the flow integral for subsystem 2 computed at step 2. Since the objective function in LPP (5.2) is minimized, there must exist $b \in B^{(1,2)}$ such that $\mathbf{m}_f(b) = 0$ where $\mathbf{m}_f = \mathbf{m}_0 + \mathbf{C} \cdot \begin{pmatrix} \mathbf{s}^1 \\ \mathbf{s}^2 \end{pmatrix}$. Assume that there exists $c \in B^{(2,1)}$ such that $\mathbf{m}_f(c) < 0$, then k^2 should be increased to make $\mathbf{m}_f(c)$ nonnegative. However, since there exists a P-semiflow containing b and c , the increase of k^2 would produce a negative b . Thus, it is not possible to make b and c nonnegative simultaneously. This contradicts assumption (A1), therefore, there cannot exist such a buffer c . $\square$

5.3.1 Calculation of control inputs

Once the flow integral vectors $\mathbf{s}$ of the evolution from the initial marking to the target marking have been computed by Algorithm 5.1, the value of the control actions $\mathbf{u}$ can be derived in several ways (for example applying the procedure in [55]) as long as $\mathbf{s} = \int_{\tau_a}^{\tau_b} (\mathbf{f} - \mathbf{u}) d\tau$ is satisfied where τ_a and τ_b are the initial and final time instants respectively. Remark that $\mathbf{s}$ can be seen as a firing count vector in the untimed system and the problem of finding a control law $\mathbf{u}$ is equivalent to a reachability problem: if the desired marking is reachable in the untimed net system it is reachable in the timed one with an appropriate control law if all transitions are controllable. This result is proved in [31] (Prop. 14. 3) where a procedure that executes a firing sequence of the untimed system in the timed one is also presented.

Let us explain the application of the method explained in Chapter 3 for calculation of control inputs of subsystems. In that method we assign constant controlled flows ($\mathbf{w}$) to transitions for driving system from its initial marking to target marking, through a linear trajectory by minimizing the time τ_f . So that, $\mathbf{s} = \mathbf{w} \cdot \tau_f$.

Let $\mathbf{s}^k$ be the flow integral of subsystem $k \in K$ calculated by Algorithm 5.1. Since both of LPPs (5.1) and (5.2) cover that state equation, $\mathbf{s}^k$ satisfies the equation of the straight line connecting $\mathbf{m}_0^k$ to $\mathbf{m}_f^k$, where $\mathbf{m}_0^k$ and $\mathbf{m}_f^k$ are the initial and target markings of subsystem $k \in K$, respectively. In

order to achieve flow constraints, following inequality should be provided $\forall j \in \{1, \dots, |T^k|\}$ where i satisfies $\Pi_{ji}^z \neq 0$:

$$0 \leq s_j^k \leq \lambda_j \cdot \Pi_{ji}^z \cdot \min \{m_{0i}^k, m_{fi}^k\} \cdot \tau_f^k \quad (5.3)$$

Hence the minimum time to reach from $\mathbf{m}_0^k$ to $\mathbf{m}_f^k$ is

$$\tau_f^k = \max_{j \in \{1, 2, \dots, |T^k|\}} \left\{ \frac{s_j^k}{\lambda_j \cdot \Pi_{ji}^z \cdot \min \{m_{0i}^k, m_{fi}^k\}} \right\} \quad (5.4)$$

If it is required to drive all subsystems of DcontPN from their initial marking to target marking at the same time takes τ_f t.u.:

$$\tau_f = \max_{k \in K} \{\tau_f^k\} \quad (5.5)$$

5.4 Design of a Distributed Controller for General Structures

In this section, we will extend Algorithm 5.1 where the problem has been studied for DcontPN composed by only two subsystems in order to be applicable to distributed systems containing more than two subsystems (see Figure 5.4). The only addition to Algorithm 5.1 is iterate a part of procedure, $|K| - 1$ times. By this way, Algorithm 5.2, which is a more general version of Algorithm 5.1 is obtained. Consequently, in the case of two subsystems, Algorithm 5.2 and Algorithm 5.1 are same. We assume that for each pair of directly connected subsystems, i.e., subsystems that have common channel places, (A1), (A2) and (A3) are satisfied.

Algorithm 5.2 represents the local controller that will be executed in each subsystem separately. During the initialization, the algorithm requires the initial and target markings, the token flow matrix, the input and output buffers and the initial marking of the input buffers of the corresponding subsystem. In step 1, each subsystem computes the flow integral $\mathbf{z}$ required to reach its target marking without taking into account the marking of the buffers, this step is carried out by means of the Linear Programming Problem (LPP) (5.1). Step 2 sets the number of iterations to $|K| - 1$, where $|K|$ is the number of

Algorithm 5.2. [Distributed controller of subsystem k]

1) Solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \mathbf{z} \\ \text{s.t.} \quad & \mathbf{m}_f(P^k) - \mathbf{m}_0(P^k) = \mathbf{C}^k \cdot \mathbf{z}, \\ & \mathbf{z} \geq 0 \end{aligned} \tag{5.6}$$

2) **For** Iteration=1 **to** $|K| - 1$ **do**

3) For every $p \in B^{(*,k)}$ compute

$$q_p^{req} = \left(\sum_{t \in p^\bullet} Pre(p, t) \cdot z(t) \right) - m_0(p)$$

4) For all $p \in B^{(*,k)}$ send q_p^{req} to the connected subsys.

5) For all $p \in B^{(k,*)}$ receive r_p^{req} from the conn. subsys.

6) For all $p \in B^{(k,*)}$ compute

$$h_p = \left(\sum_{t \in \bullet p} Post(p, t) \cdot z(t) \right) - r_p^{req}$$

7) **If** $\min_{p \in B^{(k,*)}} \{h_p\} < 0$ **then** solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \mathbf{s} \\ \text{s.t.} \quad & \mathbf{m}_f(P^k) - \mathbf{m}_0(P^k) = \mathbf{C}^k \cdot \mathbf{s}, \\ & \left(\sum_{t \in \bullet p} Post(p, t) \cdot s(t) \right) \geq r_p^{req}, \forall p \in B^{(k,*)} \\ & \mathbf{s} \geq 0 \end{aligned} \tag{5.7}$$

Else

$$\mathbf{s} = \mathbf{z}$$

End_If

8) $\mathbf{z} = \mathbf{s}$

9) **End_For**

10) **return** $\mathbf{s}$

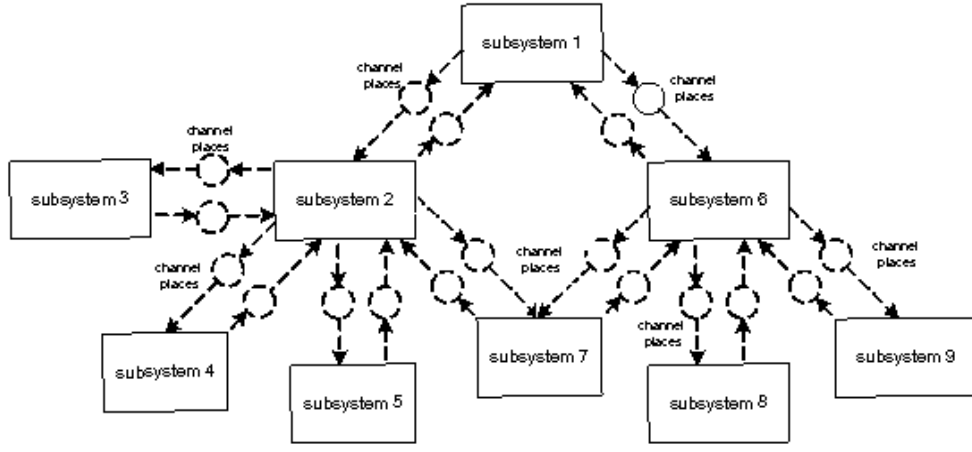


Figure 5.4: A general structure for DcontPN system

subsystems, these iterations will be used in the next section where DcontPN with general structure are considered. Step 3 computes the amounts of tokens q_p^{req} to be produced in each input buffer p in order to be able to fire z . The connected subsystems are informed about the amounts of required tokens q_p^{req} in step 4. In step 5, each subsystem receives the amount of tokens it has to produce (if any) in its output buffers.

In step 6, it is computed how many tokens would remain in each output buffer if the present control was applied. If this value is negative, more tokens must be produced in the output buffers, and therefore the control law must be recomputed. This re-computation is achieved in step 7 using LPP (5.7). Observe that comparing with LPP (5.6) of step 1 only one extra constraint is added in order to ensure that enough tokens are produced in the output buffers. Notice that, if $|K| = 2$, the algorithm only executes one iteration.

Theorem 5.2 shows that Algorithm 5.2 computes a control law for all subsystems that ensures the reachability of their target markings. Moreover, there exists a (finite) time instant at which all subsystems are in their target markings.

Theorem 5.2. *Let $\mathcal{N}$ be a DcontPN satisfying assumptions (A1), (A2) and (A3) by each pair of connected subsystems. Let $\mathbf{s}^k$ be the flow integral vectors computed by Algorithm 5.2 for subsystem k for a given initial and target marking. The application of $\mathbf{s}^k$ drives the subsystems to their target markings.*

Proof. Let us assume that Algorithm 5.2 is applied to all subsystems. Let us consider the first iteration of the algorithm. In step 1 each controller k computes the required firing count vector $\mathbf{z}^k$ to reach its final marking by solving LPP (5.6). In step 5, subsystem k , receives from all its neighbors their requirements r_p^{req} . If it is necessary, the updating of $\mathbf{z}^k$ to satisfy such requirements is achieved by LPP (5.7) in step 7.

Observe that after each iteration, the requirements of each subsystem are satisfied by the updating of the firing count vector of the neighbors subsystems what makes the marking of at least one buffer equal to zero (this is result of minimizing the firing count vector in LPP (5.7)).

Assume that the Algorithm 5.2 has been executed for $|K| - 1$ iterations and that after the last iteration, step 3 to 6 are further executed in order to obtain values for h_p . If all h_p are non-negative then the computed firing count vectors ensure the reachability of the target marking in each subsystem.

On the other hand, let us assume that at least one h_p is negative. We will prove that this leads to the global non-reachability, violating assumption (A1). An negative h_p implies that: a) at each iteration of the algorithm at least one subsystem updated its firing count vector; b) after $|K| - 1$ iterations, at least one subsystem would still need to update its firing count vector. These both facts imply that there exists a subsystem Q that would need to perform two updates, what in turn involves that there is a cycle of subsystems containing Q whose marking in the connecting buffers is not positive. Notice that there exists a P-semiflow containing such connecting buffers and places of the subsystems for which the target marking is fixed. For this reason, the target marking under consideration violates at least one existing conservation law of the conservative global system, and therefore such a marking is not globally reachable. $\square$

Example 5.8. *Let us consider DcontPN in Figure 5.5 which is composed by 4 subsystems connected in a line structure. These subsystems communicate*



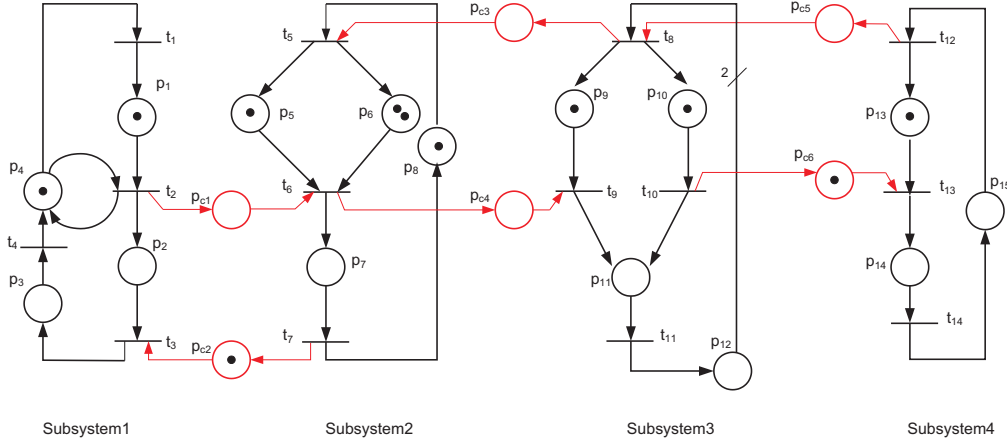


Figure 5.5: A DcontPN used in Example 5.8.

through 6 channel places. The set of buffers are: $B^{(1,2)} = \{p_{c1}\}$, $B^{(2,1)} = \{p_{c2}\}$, $B^{(3,2)} = \{p_{c3}\}$, $B^{(2,3)} = \{p_{c4}\}$, $B^{(4,3)} = \{p_{c5}\}$ and $B^{(3,4)} = \{p_{c6}\}$.

Let us assume the following initial markings: $\mathbf{m}_0(P^1) = [1 \ 0 \ 0 \ 1]^T$, $\mathbf{m}_0(P^2) = [1 \ 2 \ 0 \ 1]^T$, $\mathbf{m}_0(P^3) = [1 \ 1 \ 0 \ 0]^T$, $\mathbf{m}_0(P^4) = [1 \ 0 \ 0]^T$, $m_0(p_{c2}) = m_0(p_{c6}) = 1$, $m_0(p_{c1}) = m_0(p_{c3}) = m_0(p_{c4}) = m_0(p_{c5}) = 0$ and final markings $\mathbf{m}_f(P^1) = [1 \ 1 \ 0 \ 0]^T$, $\mathbf{m}_f(P^2) = [0 \ 1 \ 2 \ 0]^T$, $\mathbf{m}_f(P^3) = [0 \ 1 \ 1 \ 0]^T$, $\mathbf{m}_f(P^4) = [0 \ 0 \ 1]^T$.

In the first iteration of the algorithm it holds that $h_{pc1} < 0$ and $h_{pc3} < 0$, thus the flow integral of subsystem 1 and subsystem 3 are recomputed. In the second iteration, $h_{pc5} < 0$ is obtained, hence the flow integral of subsystem 4 is updated. The execution of Algorithm 5.2 is given in Table 5.3. According to final flow integrals, the final markings of the buffers are $m_f(p_{c1}) = m_f(p_{c2}) = m_f(p_{c3}) = m_f(p_{c4}) = m_f(p_{c5}) = m_f(p_{c6}) = 0$.

5.5 Case study

Let us consider the DcontPN sketched in Figure 5.6 (taken from [56]) which models two manufacturing processes containing assembly operations. The system can perform two types of products with two subsystems. These subsystems communicate through 6 buffers. The set of input (output) buffers of subsystem

Table 5.3: Execution of Algorithm 5.2 on subsystems of DcontPN in Figure 5.5

Step	Subsystem 1	Subsystem 2	Subsystem 3	Subsystem 4
Step 1	$\mathbf{z} = [1 \ 1 \ 0 \ 0]^T$	$\mathbf{z} = [1 \ 2 \ 0]^T$	$\mathbf{z} = [0 \ 1 \ 0 \ 0]^T$	$\mathbf{z} = [0 \ 1 \ 1]^T$
Step 2	<i>Iteration = 1</i>	<i>Iteration = 1</i>	<i>Iteration = 1</i>	<i>Iteration = 1</i>
Step 3 & 4	Send $q_{pc2}^{req} = -1$	Send $q_{pc1}^{req} = 2$ Send $q_{pc3}^{req} = 1$	Send $q_{pc4}^{req} = 1$ Send $q_{pc5}^{req} = 0$	Send $q_{pc6}^{req} = 0$
Step 5	Receive $r_{pc1}^{req} = 2$	Receive $r_{pc4}^{req} = 1$ Receive $r_{pc2}^{req} = -1$	Receive $r_{pc6}^{req} = 0$ Receive $r_{pc3}^{req} = 1$	Receive $r_{pc5}^{req} = 0$
Step 6	$h_{pc1} = -1$	$h_{pc2} = h_{pc4} = 1$	$h_{pc3} = -1, h_{pc6} = 0$	$h_{pc5} = 0$
Step 7	$\mathbf{s} = [2 \ 2 \ 1 \ 1]^T$	$\mathbf{s} = [1 \ 2 \ 0]^T$	$\mathbf{s} = [1 \ 2 \ 1 \ 2]^T$	$\mathbf{s} = [0 \ 1 \ 1]^T$
Step 8	$\mathbf{z} = \mathbf{s}$	$\mathbf{z} = \mathbf{s}$	$\mathbf{z} = \mathbf{s}$	$\mathbf{z} = \mathbf{s}$
Step 2	<i>Iteration = 2</i>	<i>Iteration = 2</i>	<i>Iteration = 2</i>	<i>Iteration = 2</i>
Step 3 & 4	Send $q_{pc2}^{req} = 0$	Send $q_{pc1}^{req} = 2$ Send $q_{pc3}^{req} = 1$	Send $q_{pc4}^{req} = 1$ Send $q_{pc5}^{req} = 1$	Send $q_{pc6}^{req} = 1$
Step 5	Receive $r_{pc1}^{req} = 2$	Receive $r_{pc4}^{req} = 2$ Receive $r_{pc2}^{req} = 0$	Receive $r_{pc6}^{req} = 1$ Receive $r_{pc3}^{req} = 1$	Receive $r_{pc5}^{req} = 1$
Step 6	$h_{pc1} = 0$	$h_{pc2} = h_{pc4} = 0$	$h_{pc3} = 0, h_{pc6} = 1$	$h_{pc5} = -1$
Step 7	$\mathbf{s} = [2 \ 2 \ 1 \ 1]^T$	$\mathbf{s} = [1 \ 2 \ 0]^T$	$\mathbf{s} = [1 \ 2 \ 1 \ 2]^T$	$\mathbf{s} = [1 \ 2 \ 2]^T$

1 (subsystem 2) is $B^{(2,1)} = \{p_{c1}, p_{c3}, p_{c5}\}$, and the set of input (output) buffers of subsystem 1 (subsystem 2) is $B^{(1,2)} = \{p_{c2}, p_{c4}, p_{c6}\}$.

Let the initial marking of subsystem 1 be $\mathbf{m}_0(P^1) = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 5\ 0\ 0\ 1\ 0\ 3]^T$, of subsystem 2 be $\mathbf{m}_0(P^2) = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 3\ 1\ 3]^T$, and $m_0(p_{c1}) = 1$, $m_0(p_{c2}) = 0$, $m_0(p_{c3}) = 1$, $m_0(p_{c4}) = 1$, $m_0(p_{c5}) = 1$, $m_0(p_{c6}) = 0$. Assume the target markings of the subsystems: $\mathbf{m}_f(P^1) = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 5\ 0\ 1\ 0\ 0\ 1]^T$ and $\mathbf{m}_f(P^2) = [0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 3\ 1\ 2]^T$.

Let us execute the control Algorithm 5.2. At the side of subsystem 1, $z(t_1) = z(t_7) = z(t_8) = 1$, $z(t_2) = z(t_3) = z(t_4) = z(t_5) = z(t_6) = z(t_9) = 0$ are calculated, yielding $q_{p_{c1}}^{req} = 0$, $q_{p_{c3}}^{req} = 0$, $q_{p_{c5}}^{req} = -1$. At the side of subsystem 2, $z(t_{10}) = z(t_{11}) = z(t_{12}) = z(t_{14}) = z(t_{15}) = 1$, $z(t_{16}) = z(t_{17}) = z(t_{18}) = 0$, $z(t_{13}) = 2$ are calculated, yielding $q_{p_{c2}}^{req} = 1$, $q_{p_{c4}}^{req} = 1$, $q_{p_{c6}}^{req} = 0$.

According to these values, $h_{p_{c2}} = -1$, $h_{p_{c4}} = -1$, $h_{p_{c6}} = 0$ is obtained at step 5 for the output buffers of subsystem 1. Hence, it is necessary to recompute the flow integrals. With this re-computation the final values are: $s(t_1) = s(t_7) = s(t_8) = 2$, $s(t_2) = s(t_3) = s(t_4) = s(t_5) = s(t_6) = s(t_9) = 1$.

However, for the output buffers of subsystem 2 we have $h_{p_{c1}} = 1$, $h_{p_{c3}} = 1$, $h_{p_{c5}} = 1$. Hence it is not needed to recalculate flow integrals at this side. The final values of flow integrals are same as the first calculation. By using these final control law, $\mathbf{m}_f(P^1)$, $\mathbf{m}_f(P^2)$ can be reached and the final markings of buffers are $m_f(p_{c1}) = 0$, $m_f(p_{c2}) = 0$, $m_f(p_{c3}) = 0$, $m_f(p_{c4}) = 0$, $m_f(p_{c5}) = 1$, $m_f(p_{c6}) = 1$.

5.6 Summary

Distributed systems are composed of several subsystems that exchange parts in order to obtain a given goal. This work focus on distributed systems modeled by timed continuous Petri nets (DcontPN). DcontPN is composed of several subsystems which are interconnected through channels modeled by places. This work considers the control problem of driving a DcontPN from a initial state to the desired target state.

The approach developed here is based on the design of a local controller

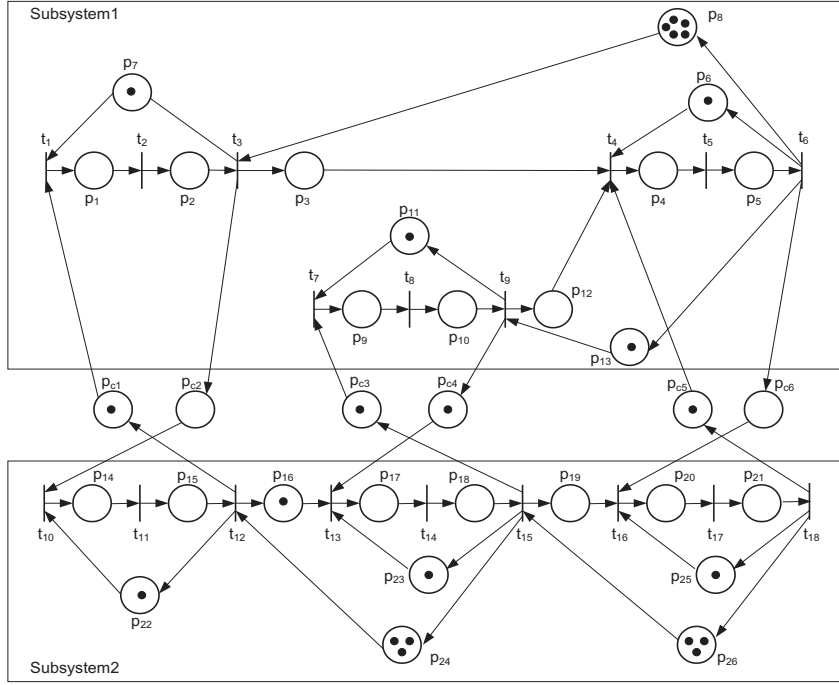


Figure 5.6: A DcontPN modeling two manufacturing processes.

for each subsystem. The main difficulties that must be taken into account when dealing with the mentioned control problem are related to the coordination among local controllers and the possibility of reaching the target marking in every subsystem.

Algorithm 5.2 is developed for control of DcontPN that consists of finite number of subsystems and it represents the local controller that will be executed in each subsystem separately. It is proved that, under certain assumptions on the system, the proposed algorithms always yields an appropriate control law. A manufacturing system was taken as case study to illustrate the control method.

6. CONCLUDING REMARKS

PNs are frequently used for modeling, analysis and synthesis of DESs. The distributed state or marking of a PN is given by a vector of natural numbers which represent the number of tokens in each place. Similar to most formalisms for DESs, PNs suffer from the state explosion problem. One possible way to overcome this problem is fluidification which has been known and used for approaching a discrete model of a system by a relaxed continuous one, for a long while. The fluidification of PNs results in continuous Petri nets. The quantity that a transition is fired is not constrained to the natural numbers but positive real numbers. Thus, markings of continuous Petri nets are vectors of nonnegative real numbers. As in PNs, time can easily be introduced to continuous Petri nets. Different server semantics can be adopted for the firing of transitions. Among them the most widely used semantics are finite servers semantics and infinite servers semantics.

This thesis mainly deals with timed continuous Petri nets with infinite server semantics, contPN , which is a particular case of PWL systems. But the control techniques related to control design of PWL systems can not be applied to contPN because of its particular dynamic input constraints. Therefore, it is necessary to investigate new control methods for contPN s.

In Chapter 3, a control strategy which aims at driving the system from an initial state to a target one through a linear trajectory by minimizing the time is developed. Then, in order to obtain faster trajectories, intermediate states, not necessarily on the line connecting the initial and the target marking are computed. Moreover, an algorithm that recursively refines the initially obtained trajectory is given. These computations are then used to develop a closed loop scheme (publications on the topic [55, 57]).

In Chapter 4, the control problem where some components of the target state are not specified is addressed. For calculating the value of unspecified components the control strategy in Chapter 3 requests solving NLP. In order to reduce computational complexity, we studied on another control strategy

and considered JF contPN as a starting point. This strategy proposes an online algorithm and requests solving BLP for calculating unspecified components.

In Chapter 5, distributed timed continuous Petri net system (DcontPN), which is composed of several subsystems interconnected through channels modeled by places, is introduced. First, the control problem of driving a DcontPN consisting of two subsystems from a initial state to a target state is considered. Then it is shown that the controllers can be easily extended to handle DcontPN structures composed by more than two subsystems (first publications on the topic [58, 59, 60]).

However, there are some issues that need to be addressed. For example, control problem considering target markings with unspecified components is studied only for JF contPN; the results on distributed control hold only for MTS net systems. An interesting point is to extend them to more general structures.

REFERENCES

- [1] J. Júlvez. *Timed Continuous Petri Nets: Quantitative Analysis, Observability and Control*. PhD thesis, University of Zaragoza, November 2004.
- [2] A. Zimmermann, D. Rodríguez, and M. Silva. A Two Phase Optimization Method for Petri Net Models of Manufacturing Systems. *Journal of Intelligent Manufacturing*, 12(5):421–432, October 2001.
- [3] M.C. Zhou, F. DiCesare, and D. Guo. Modeling and performance analysis of a resource-sharing manufacturing system using stochastic Petri nets. In *5th IEEE Symp. on Intelligent Control*, 1005–1010, Philadelphia, PA, 1990.
- [4] J. Xu, L. Recalde, and M. Silva. Tracking control of join-free timed continuous Petri net systems under infinite servers semantics. *Discrete Event Dynamic Systems*, 18(2):263–288, 2008.
- [5] X. Cao and Y. Ho. Models of discrete event dynamic systems. *IEEE Control Systems Magazine*, 69–76, 1990.
- [6] A.A. Desrochers. *Modeling and control of automated manufacturing systems*. IEEE Press, Piscataway, NJ, USA, 1990.
- [7] D. Mandrioli, A. Morzenti, M. Pezze, P. Pietro S., and S. Silva. A Petri net and logic approach to the specification and verification of real time systems. In *Formal Methods for Real Time Computing*. John Wiley & Sons Ltd.
- [8] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, 1989.
- [9] M. Silva and L. Recalde. Petri nets and integrality relaxations a view of continuous Petri nets. *IEEE Trans on Systems Man and Cybernetics*, 32(4):314–327, 2002.

- [10] A. Aybar, A. İftar, and H. Apaydın-Özkan. Centralized and decentralized supervisory controller design to enforce boundedness, liveness, and reversibility in petri nets. *International Journal of Control*, 537–553, 2005.
- [11] A. Aybar and A. İftar. Decentralized structural controller design for large-scale discrete event systems modelled by petri nets. *Kybernetika*, 1(45):3–14, 2009.
- [12] T. Nishi, K. Shimatani, and M. Inuiguchi. Lagrangian relaxation technique for solving scheduling problems by decomposition of timed petri nets. In *Proceedings of the 17th IFAC World Congress*, 17, COEX, Korea, South, 2008.
- [13] R. David and H. Alla. Continuous Petri nets. In *8th European Workshop on Application and Theory of Petri Nets*, Zaragoza, Spain, 1987.
- [14] M. Silva and L. Recalde. Continuization of timed Petri nets: From performance evaluation to observation and control. In *Proc. of the 26th Int. Conf. On Application and Theory of Petri Nets and Other Models of Concurrency*. Springer-Verlag, 2005.
- [15] A. Aybar and A. İftar. Deadlock avoidance controller design for timed petri nets using stretching. *IEEE Systems Journal*, 2:178–188, 2008.
- [16] J.L. Farges and J.L. Gallego. Stochastic p-timed petri nets and their use for traffic queue estimation. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1, 829–834, Tokyo, Japan, 1999.
- [17] G. Florin and S. Natkin. Evaluation based upon stochastic Petri nets of the maximum throughput of a full duplex protocol. In *Application and theory of Petri nets: selected papers from the first and the second European workshop*, 280–288. Springer-Verlag, Berlin, Germany, 1982.
- [18] M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. PhD thesis, University of California, 1981.

- [19] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*.
- [20] B. Gaujal and A. Giua. Optimal stationary behavior for a class of timed continuous Petri nets. *IEEE Trans. on Robotics and Automation*, 40(9):1505–1516, 2004.
- [21] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8:159–188, 1998.
- [22] R. David and H. Alla. *Autonomous and timed continuous Petri nets*. Springer, Berlin, 2005.
- [23] M. Silva and L. Recalde. On fluidification of Petri net models: from discrete to hybrid and continuous models. *Annual Reviews in Control*, 28(2):253–266, 2004.
- [24] C. Mahulea, L. Recalde, and M. Silva. On performance monotonicity and basic servers semantics of continuous Petri nets. In *8th International Workshop on Discrete Event Systems*, 345–351, Ann Arbor, USA, July 2006.
- [25] E. Sotang. An algebraic approach to bounded controllability of linear systems. *International Journal of Control*.
- [26] H. Sussmann, E. Sontag, and Y. Yang. A General Result on the Stabilization of Linear Systems using Bounded Controls. In *Proceedings of the 32nd Conference on Decisions and control*, 1802–1807, San Antonio, USA, 1993.
- [27] R. Brammer. Controllability in linear autonomous systems with positive controllers. *SIAM Journal Control*, 10(2):329353, 1972.
- [28] C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. Optimal model predictive control of timed continuous Petri nets. *IEEE Transactions on Automatic Control*, 53(7):1731 – 1735, August 2008.

- [29] J. Xu, L. Recalde, and M. Silva. Tracking control of timed continuous Petri net systems under infinite servers semantics. In *17th World Congress of IFAC*, 3192–3197, Seoul, Korea, 2008.
- [30] C.R. Vázquez and M. Silva. Piecewise-linear constrained control for timed continuous petri nets. In *IEEE Conference on Decision and Control*, Shanghai, China, 12/2009 2009.
- [31] C. Mahulea, A. Ramirez, L. Recalde, and M.Silva. Steady state control reference and token conservation laws in continuous Petri net systems. *IEEE Transactions on Automation Science and Engineering*, 5(2):307–320, April 2008.
- [32] C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. On sampling continuous timed Petri nets: reachability equivalence under infinite servers semantics. 37–43, 6 2006.
- [33] J. Komenda and J. H. van Schuppen. Control of discrete-event systems with modular or distributed structure. *Theoretical Computer Science*, 388(1):199–226, 2007.
- [34] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete-event systems. *Mathematics of Control, Signals, and Systems*, 1(1):13–30, 1988.
- [35] P. Krishnan. Distributed timed automata. *Electronic Notes in Theoretical Computer Science*, 2000.
- [36] P. Darondeau. Distributed implementation of ramadge-wonham supervisory control with Petri nets. In *IEEE Conference on Decision and Control*, 2107–2112, December 2005.
- [37] R. P. Moreno, D. Tardioli, and J.I.V.Salcedo. Distributed implementation of discrete event control systems based on Petri nets. In *ISIE08:IEEE International Symposium on Industrial Electronics*, 1738–1745, June 2008.

- [38] B. Bordbar, L. Giacomani, and D.J. Holding. Design of distributed manufacturing systems using uml and Petri nets. In *Proc. of 6th IFAC Workshop on Algorithms and Architectures for Real-Time Control*, 91–96, Mallorca, Spain, May 2000.
- [39] E. Fabre and L. Jezequel. Distributed Optimal Planning: An Approach by Weighted Automata Calculus. In *48th Conference on Decision and Control*, 211 –216, Shanghai, P.R. China, December 2009.
- [40] J. Júlvez, L. Recalde, and M. Silva. Controllability in linear autonomous systems with positive controllers. *Automatica*, 41(4):605–616, 2005.
- [41] J. Campos and M. Silva. Structural techniques and performance bounds of stochastic Petri net models. *Lecture Notes in Computer Science*.
- [42] F. Commoner, A. W. Holt, S. Even, and A. Pnueli. Marked directed graphs. *Journal on Computer Systems Science*, 5:72–79, 1971.
- [43] E. Teruel, J. M. Colom, and M. Silva. Choice-free Petri nets: A model for deterministic concurrent systems with bulk services and arrivals. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(27):73–83, 1997.
- [44] M. H. T. Hack. *Analysis of production schemata by Petri nets*. PhD thesis, M.I.T., Cambridge, MA, USA, 1972.
- [45] C. Mahulea, L. Recalde, and M. Silva. Basic server semantics and performance monotonicity of continuous Petri nets. *Discrete Event Dynamic Systems*, 19:212, 06/2 2009.
- [46] L. Recalde, E. Teruel, and M. Silva. On linear algebraic techniques for liveness analysis of P/T systems. *Journal of Circuits Systems and Computers*, 8(1):223–265, 1998.
- [47] C. Mahulea. *Timed Continuous Petri Nets: Quantitative Analysis, Observability and Control*. PhD thesis, University of Zaragoza, September 2007.

- [48] L. Recalde, E. Teruel, and M. Silva. Autonomous continuous P/T systems. In S. Donatelli and J. Kleijn, editors, *Application and Theory of Petri Nets*, 1639, 107–126. Springer, 1999.
- [49] J. Júlvez, L. Recalde, and M. Silva. On reachability in autonomous continuous Petri net systems. In *24th International Conference on Application and Theory of Petri Nets*, 221–240, Eindhoven, Netherlands, June 2003. Springer.
- [50] L. Recalde and M. Silva. Petri nets fluidification revisited semantics and steady state. *APII JESA*, 35(4):435–449, 2001.
- [51] R. David and H. Alla. Modeling of production systems by continuous Petri nets. In *International Conference of CAD/CAM, Robotics & Factories*, Southfield, U.K., August 1988.
- [52] R. David, X. Xie, and Y. Dallery. Properties of Continuous Models of Transfer Lines with Unreliable Machines and Finite Buffers. *IMA. Journal of Mathematics Applied in Business & Industry*, 2(4):281–308, 1989.
- [53] J. Júlvez, L. Recalde, and M. Silva. Deadlock-freeness analysis of continuous mono-T-semiflow Petri nets. *IEEE Trans. on Automatic Control*, 51(9):1472–1481, 2006.
- [54] J. Júlvez, L. Recalde, and M. Silva. Steady-state performance evaluation of continuous mono-T-semiflow Petri nets. *Automatica*, 41(4):605–616, 2005.
- [55] H. Apaydın-Özkan, J. Júlvez, C. Mahulea, and M. Silva. An efficient heuristics for minimum time control of continuous Petri nets. In *3rd IFAC Conference on Analysis and Design of Hybrid Systems*, 44–49, Zaragoza, Spain, September 2009.
- [56] J.M. Proth. Petri nets for modelling and evaluating deterministic and stochastic manufacturing systems. In *PNPM '97: Proceedings of the 6th*

International Workshop on Petri Nets and Performance Models, 3192–3197, Washington, DC, USA, 1997.

- [57] H. Apaydın-Özkan, J. Júvez, C. Mahulea, and M. Silva. Approaching minimum time control of timed continuous Petri nets. *Nonlinear Analysis: Hybrid Systems*, 2010 (in press).
- [58] H. Apaydın-Özkan, J. Júvez, C. Mahulea, and M. Silva. A Control Method for Timed Distributed Continuous Petri nets. In *2010 American Control Conference*, Baltimore, USA, June 2010 (to appear).
- [59] H. Apaydın-Özkan, C. Mahulea, J. Júvez, and M. Silva. An iterative control method for distributed continuous Petri nets with tree structure. In *In 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, USA, December 2010.
- [60] H. Apaydın-Özkan, C. Mahulea, J. Júvez, and M. Silva. On control of distributed continuous Petri nets. Research report, University of Zaragoza (Submitted for publication), 2010.

APPENDIX A

A.1. Matlab code for Linear Trajectory

```
%segment1

m0=[7.5-1/6 4.5+1/6 4+1/3 2-1/3 1.5-1/6 5-1/3 4+1/3 2.5+1/6]';
mf=[7 5 5 1 1 4 5 3];
mf1=mf(1);mf2=mf(2);mf3=mf(3);mf4=mf(4);
mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);

Aeq=[
-1 0 0 1 0
1 -1 0 0 0
0 1 -1 0 0
0 0 1 -1 0
-1 1 0 0 0
0 -1 1 0 0
0 0 -1 1 0
-1 1 -1 1 0];
beq=[mf1-m01;mf2-m02;mf3-m03;mf4-m04
mf5-m05;mf6-m06;mf7-m07;mf8-m08];

A=[
1 0 0 0 -4*min(m05,mf5)
0 1 0 0 -min(m02,mf2)
0 0 1 0 -3*min(m08,mf8)
0 0 0 1 -min(m04,mf4)];
b=zeros(4,1)
```

```

lb=zeros(5,1);
f=[0 0 0 0 1]
x = linprog(f,A,b,Aeq,beq,lb,[]);

%%segment2

m0=[7.5 4.5 4 2 1.5 5 4 2.5];
mf=[7.5-1/6 4.5+1/6 4+1/3 2-1/3 1.5-1/6 5-1/3 4+1/3 2.5+1/6]';
mf1=mf(1); mf2=mf(2);mf3=mf(3);mf4=mf(4);
mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);

Aeq=[
-1 0 0 1 0
1 -1 0 0 0
0 1 -1 0 0
0 0 1 -1 0
-1 1 0 0 0
0 -1 1 0 0
0 0 -1 1 0
-1 1 -1 1 0];
beq=[mf1-m01;mf2-m02;mf3-m03;mf4-m04
mf5-m05;mf6-m06;mf7-m07;mf8-m08];

A=[
1 0 0 0 -4*min(m05,mf5)
0 1 0 0 -min(m02,mf2)
0 0 1 0 -3*min(m08,mf8)
0 0 0 1 -min(m04,mf4)];
b=zeros(4,1)

```

```

lb=zeros(5,1);
x0=zeros(20,1);
f=[0 0 0 0 1]
x = linprog(f,A,b,Aeq,beq,lb,[]);

```

A.2. Matlab code for PiecewiseLinear Trajectory

```
%intermediate state on the border
```

```
m0=[7.5 4.5 4 2 1.5 5 4 2.5];
```

```
mf=[7 5 5 1 1 4 5 3];
```

```
Path_0=[m0;mf];
```

```
Aeq=[
```

```

1 0 0 1 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0
1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
-1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
0 -1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 0 -1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0
-1 1 -1 1 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0
0 0 0 0 -1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 -1 1 -1 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 -1 0 0 0 0 0

```

```

0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0];
beq=[-m0(1);-m0(2);-m0(3);-m0(4);-m0(5);
      -m0(6);-m0(7);m0(8);mf(1);mf(2);mf(3);
      mf(4);mf(5);mf(6);mf(7);mf(8);0;6;9;9
      18;6];

A=[
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0];
b=[7.5 ; -7; 5 ; -4.5 ; 5 ; -4 ; 2 ; -1 ; 1.5
    -1 ; 5 ; -4 ; 5 ; -4 ; 3 ; -2.5 ];

lb=zeros(18,1);
x0=ones(18,1);
x0=rand(18,1);%[0 1 1 0 1 0 0 1 0 3 1.5 1 1]';

```

```
[x,fval] = fmincon(@min_border,x0,A,b,Aeq,beq,lb,[],@fun_border);
t=x(17)+x(18);
md=x(9:16)
```

```
function [c,ceq]=fun_border(x)
c(1)=x(1)-4*x(13)*x(17);
c(2)=x(2)-4.5*x(17);
c(3)=x(3)-7.5*x(17);
c(4)=x(4)-x(12)*x(17);
c(5)=x(5)-4*x(18);
c(6)=x(6)-x(18);
c(7)=x(7)-3*x(16)*x(18);
c(8)=x(8)-x(18);
ceq=[];
end
```

```
function f = min_border(x)
f = x(18) + x(17);
```

```
%Interior Intermediate States
```

```
m0=[7.5 4.5 4 2 1.5 5 4 2.5];
mf=md;
mf1=mf(1);mf2=mf(2);mf3=mf(3);mf4=mf(4);
mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
```

```
A=[
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
```

```

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0];
b=[m01;(-mf1);mf2;(-m02);mf3;(-m03);m04;(-mf4);
    m05;(-mf5);m06;(-mf6);mf7;(-m07);mf8;(-m08)];

```

```

Aeq=[
-1 0 0 1 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
-1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 -1 1 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 0 -1 1 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
-1 1 -1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0
0 0 0 0 -1 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0
0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0 0

```

```

0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0
0 0 0 0 -1 1 -1 1 0 0 0 0 0 0 0 1 0 0];
beq=[(-m01);(-m02);(-m03);(-m04);(-m05);(-m06);(-m07); (m08);
      mf1;mf2;mf3;mf4;mf5;mf6;mf7;mf8;6;9;6;9;18];

lb=zeros(18,1); f=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1];
x0=zeros(18,1);

[x,fval] = fmincon(@min_int1,x0,A,b,Aeq,beq,lb,[],@fun_int1);

function [c,ceq]=fun_int1(x)
m0=[7.5;4.5;4;2;1.5;5;4;2.5];
mf=[7.5;4.5;4.5;1.5;1.5;4.5;4.5;3];
mf1=mf(1);mf2=mf(2);mf3=mf(3);mf4=mf(4);
mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
c(1)=x(1)-4*x(13)*x(17);
c(2)=x(2)-m02*x(17);
c(3)=x(3)-3*m08*x(17);
c(4)=x(4)-x(12)*x(17);
c(5)=x(5)-4*mf5*x(18);
c(6)=x(6)-x(10)*x(18);
c(7)=x(7)-3*x(16)*x(18);
c(8)=x(8)-mf4*x(18); ceq=[];
end

function f = min_int1(x)
f = x(18) + x(17);

```

%%

m0=md;

mf=[7 5 5 1 1 4 5 3];

mf1=mf(1);mf2=mf(2);mf3=mf(3);mf4=mf(4);

mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);

m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);

m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);

A=[

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0];

b=[m01;(-mf1);mf2;(-m02);mf3;(-m03);m04;(-mf4)

m05;(-mf5);m06;(-mf6);mf7;(-m07);mf8;(-m08)];

```

Aeq=[
-1 0 0 1 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
-1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
0 -1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0
0 0 -1 1 0 0 0 0 0 0 0 0 0 0 -1 0 0 0
-1 1 -1 1 0 0 0 0 0 0 0 0 0 0 0 -1 0 0
0 0 0 0 -1 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 -1 0 0 0 1 0 0 0 0 0 0
0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 -1 1 0 0 0 0 0 0 1 0 0 0
0 0 0 0 -1 1 -1 1 0 0 0 0 0 0 0 1 0 0];

beq=[(-m01);(-m02);(-m03);(-m04);(-m05);(-m06);(-m07);(-m08)
mf1;mf2;mf3;mf4;mf5;mf6;mf7;mf8;6;9;6;9;18];

lb=zeros(18,1);
f=[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1];
x0=rand(18,1);
[x,fval] = fmincon(@min_int2,x0,A,b,Aeq,beq,lb,[],@fun_int2);

function [c,ceq]= fun_int2 (x)
m0=[7.5 4.5 4.5 1.5 1.5 4.5 4.5 3];
mf=[7 5 5 1 1 4 5 3];

mf1=mf(1);mf2=mf(2);mf3=mf(3);mf4=mf(4);

```

```

mf5=mf(5);mf6=mf(6);mf7=mf(7);mf8=mf(8);
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
c(1)=x(1)-4*x(13)*x(17);
c(2)=x(2)-x(14)*x(17);
c(3)=x(3)-3*m08*x(17);
c(4)=x(4)-x(12)*x(17);
c(5)=x(5)-4*mf5*x(18);
c(6)=x(6)-mf6*x(18);
c(7)=x(7)-3*x(16)*x(18);
c(8)=x(8)-mf4*x(18);
ceq=[];
end

```

A.3. Matlab code for Closed Loop Control

```

% segment 1

m0=[7.5 4.5 4 2 1.5 5 4 2.5];
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
ms=[7.5 4.5 4.26 1.74 1.5 4.74 4.26 2.76]
ms1=ms(1);ms2=ms(2);ms3=ms(3);ms4=ms(4);
ms5=ms(5);ms6=ms(6);ms7=ms(7);ms8=ms(8);

I=[];error=1;

while error<0.1;

t=0.01;

Aeq=[-t 0 0 t (-ms1+m01)

```

```
t -t 0 0 (-ms2+m02)
0 t -t 0 (-ms3+m03)
0 0 t -t (-ms4+m04)
-t t 0 0 (-ms5+m05)
0 -t t 0 (-ms6+m06)
0 0 -t t (-ms7+m07)
-t t -t t (-ms8+m08)];
beq=[0;0;0;0;0;0;0;0]
```

```
A=[
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 (1.74-m04)
0 0 0 0 1];
b=[6
4.5
3*m08
m04
1];
```

```
lb=[0;0;0;0;0];
f=[0;0;0;0;-1];
x=linprog(f,A,b,Aeq,beq,lb,[]);
alpha=x(5);
mx=alpha*ms+(1-alpha)*m0;
I=[I
k x(1) x(2) x(3) x(4)]

m0=mx;
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
```

```

m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
e1=ms(1)-mx(1);e2=ms(2)-mx(2);
e3=ms(3)-mx(3);e4=ms(4)-mx(4);
e5=ms(5)-mx(5);e6=ms(6)-mx(6);
e7=ms(7)-mx(7);e8=ms(8)-mx(8);

error=e1+e2+e3+e4+e4+e5+e6+e7+e8;

end

% segment 2

m0=[7.5 4.5 4.26 1.74 1.5 4.74 4.26 2.76];
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
ms=[7.5 4.5 4.5 1.5 1.5 4.5 4.5 3];
ms1=ms(1);ms2=ms(2);ms3=ms(3);ms4=ms(4);
ms5=ms(5);ms6=ms(6);ms7=ms(7);ms8=ms(8);

I=[];error=1;

while error>0.01;
t=0.01

Aeq=[-t 0 0 t (-ms1+m01)
t -t 0 0 (-ms2+m02)
0 t -t 0 (-ms3+m03)
0 0 t -t (-ms4+m04)
-t t 0 0 (-ms5+m05)
0 -t t 0 (-ms6+m06)
0 0 -t t (-ms7+m07)
-t t -t t (-ms8+m08)];

```

```
beq=[0;0;0;0;0;0;0;0;0]
```

```
A=[
```

```
1 0 0 0 0
```

```
0 1 0 0 0
```

```
0 0 1 0 0
```

```
0 0 0 1 1.5-m04
```

```
0 0 0 0 1];
```

```
b=[6
```

```
4.5
```

```
3*m08
```

```
m04
```

```
1];
```

```
lb=[0;0;0;0;0];
```

```
f=[0;0;0;0;-1];
```

```
x=linprog(f,A,b,Aeq,beq,lb,[]);
```

```
alpha=x(5);
```

```
mx=alpha*ms+(1-alpha)*m0;
```

```
I=[I
```

```
k x(1) x(2) x(3) x(4)]
```

```
m0=mx;
```

```
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
```

```
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
```

```
e1=ms(1)-mx(1);e2=ms(2)-mx(2);
```

```
e3=ms(3)-mx(3);e4=ms(4)-mx(4);
```

```
e5=ms(5)-mx(5);e6=ms(6)-mx(6);
```

```
e7=ms(7)-mx(7);e8=ms(8)-mx(8);
```

```
error=e1+e2+e3+e4+e4+e5+e6+e7+e8;
```

```

end

%segment 3

m0=[7.5 4.5 4.5 1.5 1.5 4.5 4.5 3];
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
ms=[7.45 4.78 4.55 1.22 1.22 4.45 4.78 3];
ms1=ms(1);ms2=ms(2);ms3=ms(3);ms4=ms(4);
ms5=ms(5);ms6=ms(6);ms7=ms(7);ms8=ms(8);

I=[];error=1;
while error>0.01;
t=0.01;

Aeq=[-t 0 0 t (-ms1+m01)
t -t 0 0 (-ms2+m02)
0 t -t 0 (-ms3+m03)
0 0 t -t (-ms4+m04)
-t t 0 0 (-ms5+m05)
0 -t t 0 (-ms6+m06)
0 0 -t t (-ms7+m07)
-t t -t t (-ms8+m08)];
beq=[0;0;0;0;0;0;0;0]

A=[
1 0 0 0 4*(m05-ms5)
0 1 0 0 m06-ms6
0 0 1 0 0
0 0 0 1 m04-ms4
0 0 0 0 1];

```

```

b=[4*m05
    m06
    9
    m04
    1];

lb=[0;0;0;0;0];
f=[0;0;0;0;-1];
x=linprog(f,A,b,Aeq,beq,lb,[]);
alpha=x(5);
mx=alpha*ms+(1-alpha)*m0;

I=[I
    k x(1) x(2) x(3) x(4)]

ma=m0;m0=mx;
m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);
mb=mx;

e1=ms(1)-mx(1);e2=ms(2)-mx(2);
e3=ms(3)-mx(3);e4=ms(4)-mx(4);
e5=ms(5)-mx(5);e6=ms(6)-mx(6);
e7=ms(7)-mx(7);e8=ms(8)-mx(8);

error=e1+e2+e3+e4+e4+e5+e6+e7+e8;
end

%segment 4

m0=[7.45 4.78 4.55 1.22 1.22 4.45 4.78 3];

```



```

m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);
m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);

ms=[7 5 5 1 1 4 5 3];
ms1=ms(1);ms2=ms(2);ms3=ms(3);ms4=ms(4);
ms5=ms(5);ms6=ms(6);ms7=ms(7);ms8=ms(8);

I=[];error=1;

while error>0.01;
t=0.01;

Aeq=[-t 0 0 t (-ms1+m01)
t -t 0 0 (-ms2+m02)
0 t -t 0 (-ms3+m03)
0 0 t -t (-ms4+m04)
-t t 0 0 (-ms5+m05)
0 -t t 0 (-ms6+m06)
0 0 -t t (-ms7+m07)
-t t -t t (-ms8+m08)];
beq=[0;0;0;0;0;0;0;0]

A=[
1 0 0 0 4*(m05-ms5)
0 1 0 0 m06-ms6
0 0 1 0 0
0 0 0 1 m04-ms4
0 0 0 0 1];
b=[4*m05
m06
9

```

m04

1];

lb=[0;0;0;0;0];

f=[0;0;0;0;-1];

x=linprog(f,A,b,Aeq,beq,lb,[]);

alpha=x(5);

mx=alpha*ms+(1-alpha)*m0;

I=[I

k x(1) x(2) x(3) x(4)];

m0=mx

m01=m0(1);m02=m0(2);m03=m0(3);m04=m0(4);

m05=m0(5);m06=m0(6);m07=m0(7);m08=m0(8);

e1=ms(1)-mx(1);e2=ms(2)-mx(2);

e3=ms(3)-mx(3);e4=ms(4)-mx(4);

e5=ms(5)-mx(5);e6=ms(6)-mx(6);

e7=ms(7)-mx(7);e8=ms(8)-mx(8);

error=e1+e2+e3+e4+e4+e5+e6+e7+e8;

end