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

P	$\boldsymbol{m}_0(\boldsymbol{m}_f)$	$\boldsymbol{m_0}^1(\boldsymbol{m_f}^1)$	$\boldsymbol{m_0}^2(\boldsymbol{m_f}^2)$	T	$\boldsymbol{\sigma}_{min}$	$\boldsymbol{\sigma}^1_{min}$	$\boldsymbol{\sigma}^2_{min}$
p_1	0 (0.4)	0 (0.4)	0 (0.4)	t_1	1.4	0.9	1.4
p_2	0 (0.3)	0 (0.3)	0 (0.3)	t_2	0.55	0.3	0.55
p_3	0 (0.3)		0 (0.3)	t_3	1	0.5	1
p_4	0 (0.3)		0 (0.3)	t_4	0.25		0.25
p_5	1 (1.3)		1 (1.3)	t_5	0.7		0.7
p_6	0 (0.5)		0 (0.5)	t_6	0		0
p_7	1 (0.3)		1 (0.3)	t_7	1.4		1.4
p_8	1 (0.4)		1 (0.4)	t_8	1	0.5	1
p_9	0 (0.2)		0 (0.2)	t_9	0.4	0.23	0.4
p_{10}	0 (0.6)	0 (0.6)	0 (0.6)	t_{10}	0.8	0.3	0.8
p_{11}	0 (0.2)	0 (0.2)		t_{11}	0.6	0.1	
p_{12}	0 (0.1)	0 (0.1)		t_{12}	0.7	0.2	
p_{13}	0 (0.1)	0 (0.1)		t_{13}	0.35	0.1	
p_{14}	0 (0.3)	0 (0.3)		t_{14}	0.5	0	
p_{15}	0 (0.1)	0 (0.1)		t_{15}	0.6	0.1	
p_{16}	0 (0.1)	0 (0.1)		t_{16}	0.25	0	
p_{17}	1 (0.1)	1 (0.1)					
p_{18}	1 (0.2)	1 (0.2)					
p_{19}	1 (0.1)	1 (0.1)					
p_{2_8}		2 (2.1)					
p_{3_9}		1 (0.8)					
p_{10_1}			1 (0.4)				

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

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

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

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

Algorithm 6 Coordinator

```
Input: $\boldsymbol{\sigma}^i_{min}$, $\boldsymbol{x}^i$, $i = 1, 2, .., K$
Output: $\alpha^i$, $i = 1, 2, .., K$
1: Receive $\boldsymbol{\sigma}^i_{min}$ and $\boldsymbol{x}^i$ from local controllers
2: Compute $\alpha^i$ by solving LPP:
3: Send $\alpha^i$ to $\mathcal{CS}^i$;
```

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

objective function of LPP (5.2). ■

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

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

Algorithm 7 Local Controller i

```
Input: $\mathcal{CN}^i$, $\boldsymbol{m}^i_0$, $\boldsymbol{m}^i_f$
Output: $\boldsymbol{\sigma}^i$
1: Compute $\boldsymbol{\sigma}^i_{min}$ the drives the system to $\boldsymbol{m}^i_f$;
2: Compute the minimal T-semiflow $\boldsymbol{x}^i$;
3: Send $\boldsymbol{\sigma}^i_{min}$ and $\boldsymbol{x}^i$ to the coordinator;
4: Receive $\alpha^i$ from the coordinator;
5: Update $\boldsymbol{\sigma}^i \leftarrow \boldsymbol{\sigma}^i_{min} + \alpha^i \cdot \boldsymbol{x}^i$;
6: Apply the ON-OFF controller;
```

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

5.3.5 A case study

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

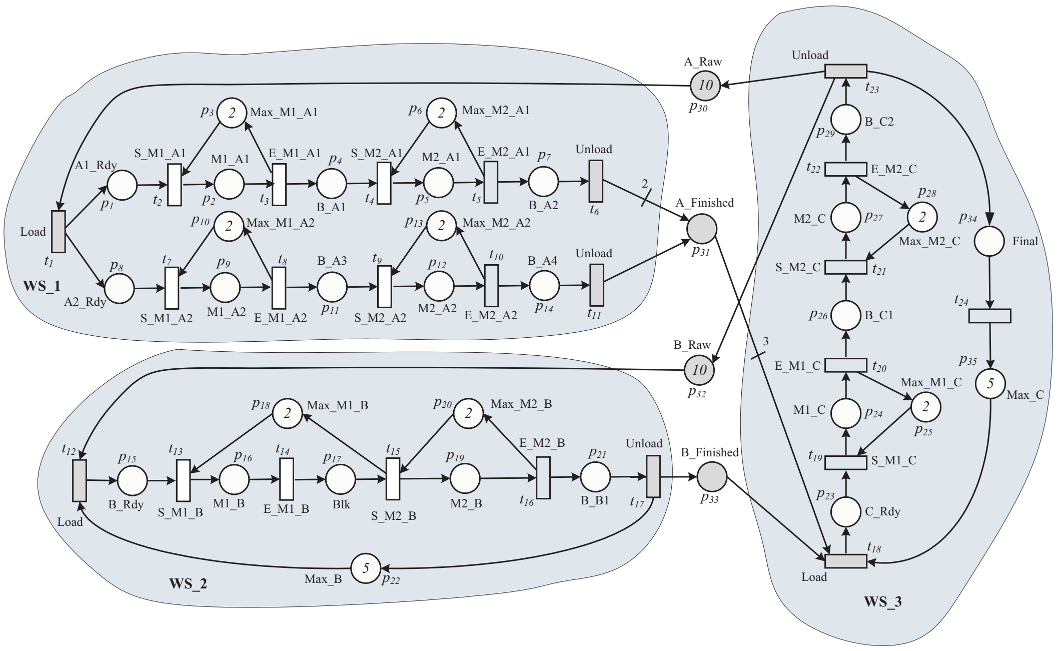


Figure 5.10: The TCPN model of a manufacturing system with three work stations.

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

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

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

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

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

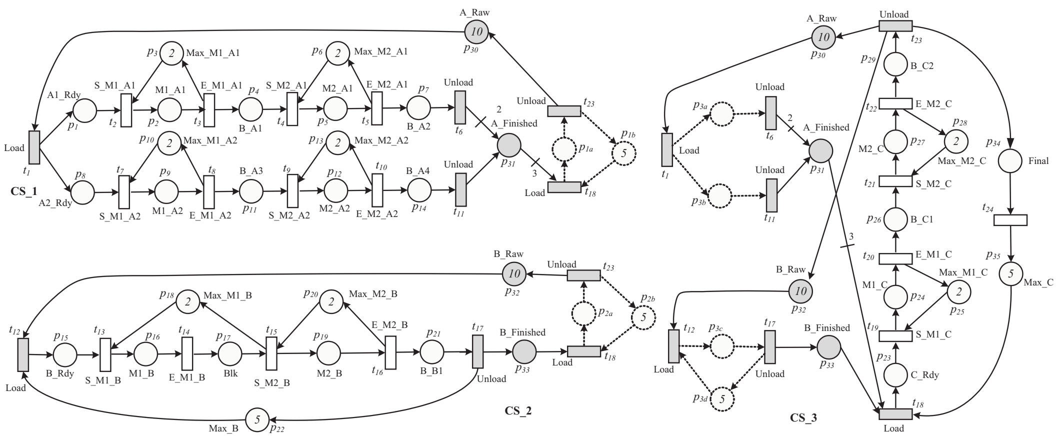


Figure 5.11: The complemented subsystems obtained from the model in Fig.5.10

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

5.4 Conclusions

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

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

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

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

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

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

Chapter 6

Distributed MPC Control of General nets

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

6.1 Introduction

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

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

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

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

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

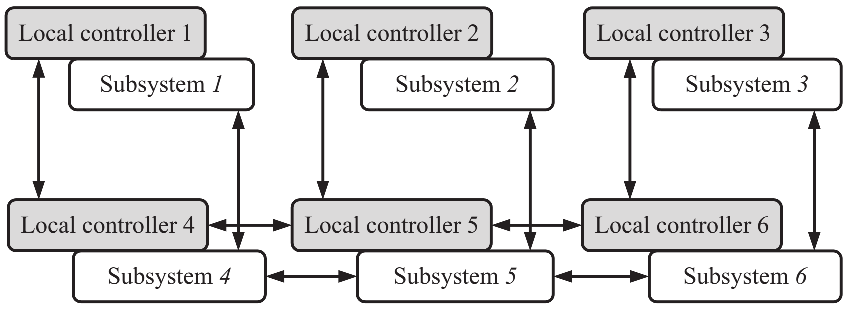


Figure 6.1: The sketch of a distributed control structure composed of 6 subsystems

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

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

6.2 A centralized MPC controller

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

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

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

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

The MPC controller is given in Algorithm 8:

Algorithm 8 A centralized MPC controller

```
Input: $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, $\boldsymbol{m}_f$, $\boldsymbol{w}_f$, $\boldsymbol{Z}$, $\boldsymbol{Q}$
Output: $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\ldots$
1: $k \leftarrow 0$;
2: while $\boldsymbol{m}_k \neq \boldsymbol{m}_f$ do
3:    Solve problem (6.2);
4:    Apply $\boldsymbol{w}_k$: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$;
5:    $k \leftarrow k + 1$;
6: end while
7: return $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\ldots$;
```

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

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

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

The cost function $J(\boldsymbol{m}_k, N)$ may be a linear or quadratic. For the target marking control problem addressed here, $J(\boldsymbol{m}_k, N)$ has the quadratic form:

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

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

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

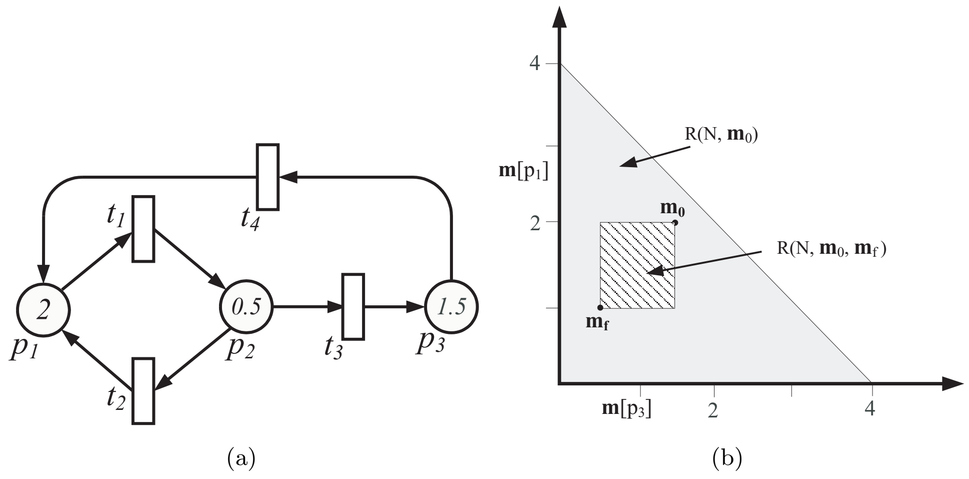


Figure 6.2: (a) A consistent and conservative CPN (a simple state machine); (b) the reachability space $R(\mathcal{N}, \boldsymbol{m}_0)$ and the subset $R(\mathcal{N}, \boldsymbol{m}_0, \boldsymbol{m}_f)$, $\boldsymbol{m}_f$ =[1 2.5 0.5]^T

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

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

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

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

So problem (6.2) is feasible.

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

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

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

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

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

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

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

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

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

Therefore, the following can be obtained:

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

According to the constraints (6.2d) and (6.2e), if $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, we must have $\boldsymbol{m}_{k+N} \neq \boldsymbol{m}_k$ and $\forall p_i \in P$, $|\boldsymbol{\Delta m}_k[p_i]| \geq |\boldsymbol{\Delta m}_{k+N}[p_i]|$. Otherwise, the system should stay in $\boldsymbol{m}_k$ at every predictive step, but obviously this cannot be an optimal solution: since the net is consistent, $\boldsymbol{m}_k > 0$ (inside $R(\mathcal{N}, \boldsymbol{m}_0, \boldsymbol{m}_f)$) and all transitions are controllable, the state marking can move in any direction [97]; in particular, it is possible to move towards $\boldsymbol{m}_f$, hence it is possible to decrease the distance to the desired marking in one step. Therefore, $J_2 - J_1 < 0$, i.e., Υ_2 is a better solution than Υ_1, implying that if $\boldsymbol{m}_k \neq \boldsymbol{m}_f$, by applying the MPC controller given in Algorithm 8 the system always starts moving towards $\boldsymbol{m}_f$ from the first predictive step. So $V(\boldsymbol{m}_k) > V(\boldsymbol{m}_{k+1})$ until $\boldsymbol{m}_k = \boldsymbol{m}_f$. ■

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

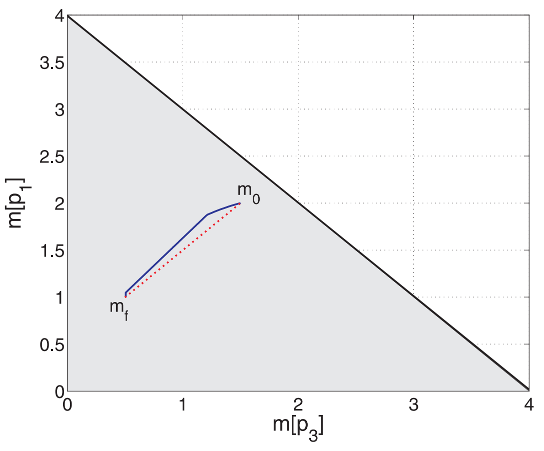


Figure 6.3: Marking trajectory of the net system in Fig. 6.2(a) by applying: the MPC controller proposed in [64] (dotted line) and the MPC controller shown in Algorithm 8 (continuous line)

6.3 Application to Distributed MPC control

Let us consider now a (large scale) PN system that is composed of a set of subsystems denoted by K. Those subsystems $S^l = \langle \mathcal{N}^l, \boldsymbol{m_0}^l \rangle \in K$ are connected with places modelling buffers denoted by B. The transitions connecting with buffer places are said to be *interface* transitions. Let P^l, T^l be the sets of places and transitions of subsystem S^l. The partition of the system is described as follows:

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

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

In the sequel, the following notations are used:

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

In the distributed setting, each subsystem is controlled independently by the MPC controller given in Algorithm 8. Therefore the states of each subsystem S^l will be interior points (inside $R(\mathcal{N}^l, \boldsymbol{m_0}^l, \boldsymbol{m_f}^l)$. However, one key issue we need to consider is that the markings of buffer places may be keeping decreased and converging to zero if the same control laws are applied (in TCPNs under infinite server semantics, once a place is marked it takes infinite to empty it). Since we consider the control in finite time, in the sequel, if $\boldsymbol{m}_k[p_i] \leq \epsilon_1$ with ϵ_1 a small positive value, we assume that under the actual control law the marking of the buffer place p_i is converging to zero. When some buffer places with their markings smaller than ϵ_1, subsystems may "stop", because the required flows to move towards the final states may be constrained by these buffer places. We will design a particular strategy to put more tokens to these places with their markings converging to zero.

6.3.1 Two subsystems

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

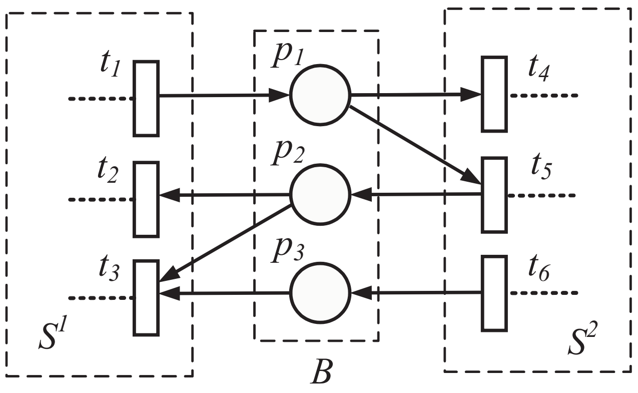


Figure 6.4: A distributed CPN composed of two subsystems

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

Let us denote by $\boldsymbol{m_0}^1$ and $\boldsymbol{m_0}^2$ the initial states (markings) of subsystems S^1 and S^2, while the desired final states are denoted by $\boldsymbol{m_f}^1$ and $\boldsymbol{m_f}^2$. The initial and final states of the global system, including the buffer places, are $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$. Instead of a global central controller, each subsystem will have its own local MPC controller. A local controller has only information about the structure and state of its corresponding subsystem, as well as the connected buffers places. The problem we address here is to drive the subsystems S^1 and S^2 to their desired final states. We use the following assumptions:

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

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

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

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

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

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

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

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

has no solution iff the continuous net system has no emptied siphon. We only need to solve problem (6.6) off-line considering the part of system composed of the buffer places and interface transitions, denoted by S': because every firing sequence that can fire in the original system S can also fire in S' and, normally S' has much smaller size than the whole system. Nevertheless, in some particular cases, it may be not necessary to solve problem (6.6). For example, given a EQ net, we can easily check its consistency and conservativeness, then by applying the rank theorems we can verify the liveness the boundedness in polynomial time. If the net system is live and bounded, then we can directly conclude that there exists no emptied siphon.

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

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

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

$$\begin{aligned}
\min \quad & J(\boldsymbol{m}_k^l, N) && \text{(6.8a)}\\
s.t.: \quad & \boldsymbol{m}_{k+j+1}^l = \boldsymbol{m}_{k+j}^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_{k+j}^l, j = 0,...,N-1 && \text{(6.8b)}\\
& \boldsymbol{G}^l \cdot \begin{bmatrix} \boldsymbol{w}_{k+j}^l \\ \boldsymbol{m}_{k+j}^l \end{bmatrix} \leq 0,\ j = 0,...,N-1 && \text{(6.8c)}\\
& \boldsymbol{w}_{k+j}^l \geq 0,\ j = 0,...,N-1 && \text{(6.8d)}\\
& {\boldsymbol{m}_f}^l[p_i] \geq \boldsymbol{m}_{k+j+1}^l[p_i] \geq \boldsymbol{m}_{k+j}^l[p_i],\ \text{if } {\boldsymbol{m}_f}^l[p_i] \geq {\boldsymbol{m}_0}^l[p_i], && \text{(6.8e)}\\
& \qquad j = 0,...,N-1 && \\
& {\boldsymbol{m}_f}^l[p_i] \leq \boldsymbol{m}_{k+j+1}^l[p_i] \leq \boldsymbol{m}_{k+j}^l[p_i],\ \text{if } {\boldsymbol{m}_f}^l[p_i] \leq {\boldsymbol{m}_0}^l[p_i], && \text{(6.8f)}\\
& \qquad j = 0,...,N-1 && \\
& \boldsymbol{w}_k^l[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_k[p_i]}{\boldsymbol{Pre}^l[p_i,t]}, \forall t \in {p_i}^\bullet, p_i \in B^{(\cdot,l)} && \text{(6.8g)}
\end{aligned}$$

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

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

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

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

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

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

$$\begin{aligned}
\min \quad & H(\boldsymbol{m}_k^l) && \text{(6.11a)}\\
s.t.: \quad & \boldsymbol{m}_{k+1}^l = \boldsymbol{m}_k^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_k^l && \text{(6.11b)}\\
& \boldsymbol{G}^l \cdot \begin{bmatrix} \boldsymbol{w}_k^l \\ \boldsymbol{m}_k^l \end{bmatrix} \leq 0 && \text{(6.11c)}\\
& \boldsymbol{w}_k^l \geq 0 && \text{(6.11d)}\\
& \boldsymbol{m_f}^l[p_i] \geq \boldsymbol{m}_{k+1}^l[p_i] \geq \boldsymbol{m}_k^l[p_i], \text{ if } \boldsymbol{m_f}^l[p_i] \geq \boldsymbol{m_0}^l[p_i] && \text{(6.11e)}\\
& \boldsymbol{m_f}^l[p_i] \leq \boldsymbol{m}_{k+1}^l[p_i] \leq \boldsymbol{m}_k^l[p_i], \text{ if } \boldsymbol{m_f}^l[p_i] \leq \boldsymbol{m_0}^l[p_i] && \text{(6.11f)}\\
& \boldsymbol{w}_k^l[t] \leq \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_k^l[p_i]}{\boldsymbol{Pre}^l[p_i, t]}, \forall t \in p_i{}^\bullet, p_i \in B^{(\cdot,l)} && \text{(6.11g)}\\
& \Theta \cdot \sum_{t_j \in p_i{}^\bullet} \boldsymbol{w}_k^l[t_j] \cdot \boldsymbol{Pre}[p_i, t_j] \leq \alpha \cdot \boldsymbol{m}_k[p_i], \forall p_i \in B^{(\cdot,l)} && \text{(6.11h)}
\end{aligned}$$

The procedure of the distributed MPC controller consists of solving problem (6.8) and/or problem (6.11) in each subsystem S^l at any time step: if the final state $\boldsymbol{m_f}^l$ has already been reached, problem (6.11) is solved, trying to put more tokens to its output buffers with their markings converging to zero (remember that S^l stays in $\boldsymbol{m_f}^l$ because of constraints (6.11e) and (6.11f)); otherwise, problem (6.8) is first solved and, if subsystem S^l is able to evolve towards its final state (still inside the convex $R(\mathcal{N}^l, \boldsymbol{m}_0^l, \boldsymbol{m}_f^l)$), the first predictive control law is applied; if by solving (6.8) the system stops in $\boldsymbol{m}_k$, then problem (6.11) is solved. Because one subsystem may reach the final state faster than the other, each subsystem should communicate to its neighbors when its final state has been reached. This procedure repeats until the final states of both subsystems have been reached. The local controller of subsystem S^l is given in Algorithm 9.

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

```
Input: $S^l$, $\boldsymbol{Z}$, $\boldsymbol{Q}$, $\boldsymbol{m_f}^l$, $\alpha$
Output: $\boldsymbol{w}_0^l$, $\boldsymbol{w}_1^l$, $\boldsymbol{w}_2^l$, $\dots$
 1: $k \leftarrow 0$;
 2: while $\exists S^i, \boldsymbol{m}_k^i \neq \boldsymbol{m_f}^i$, $i = 1,2$ do
 3:     if $\boldsymbol{m}_k^l = \boldsymbol{m_f}^l$ then
 4:         Solve problem (6.11)
 5:     else
 6:         Solve problem (6.8)
 7:         if $\boldsymbol{m}_k^l = \boldsymbol{m}_{k+1}^l$ then
 8:             Solve problem (6.11)
 9:         end if
10:     end if
11:     Apply $\boldsymbol{w}_k^l$: $\boldsymbol{m}_{k+1}^l \leftarrow \boldsymbol{m}_k^l + \Theta \cdot \boldsymbol{C}^l \cdot \boldsymbol{w}_k^l$
12:     Update the states of buffers
13:     $k \leftarrow k + 1$
14: end while
15: return $\boldsymbol{w}_0^l$, $\boldsymbol{w}_1^l$, $\boldsymbol{w}_2^l$, $\dots$
```

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

Proposition 6.3.2. *Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{\lambda}, \boldsymbol{m}_0 \rangle$ be TCPN system composed of two subsystems $\mathcal{S}^l = \langle \mathcal{N}^l, \boldsymbol{\lambda}^l, \boldsymbol{m_0}^l \rangle$, $l = 1,2$. If assumptions (A1) to (A3) are satisfied, by applying Algorithm 9, each subsystem S^l converges to its corresponding final state $\boldsymbol{m_f}^l$ in finite time.*

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

Without loss of generality, assume that subsystem S^1 has stopped in a state before reaching $\boldsymbol{m_f}^1$, then problem (6.11) should be solved in S^1. Consider now subsystem S^2, there are two cases: (i) S^2 is able to keep evolving towards $\boldsymbol{m_f}^2$ by solving problem (6.8); (ii) S^2 also stops in a state before reaching $\boldsymbol{m_f}^2$. In case (i) the final state of S^2 will be reached in finite time, then problem (6.11) should be solved; in case (ii), problem (6.11) should also be solved according to the algorithm. Therefore, we need to prove that by solving (6.11) in both subsystems, these buffer places with their marking converging to zero, will get marked.

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

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

6.3.2 Multiple subsystems

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

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

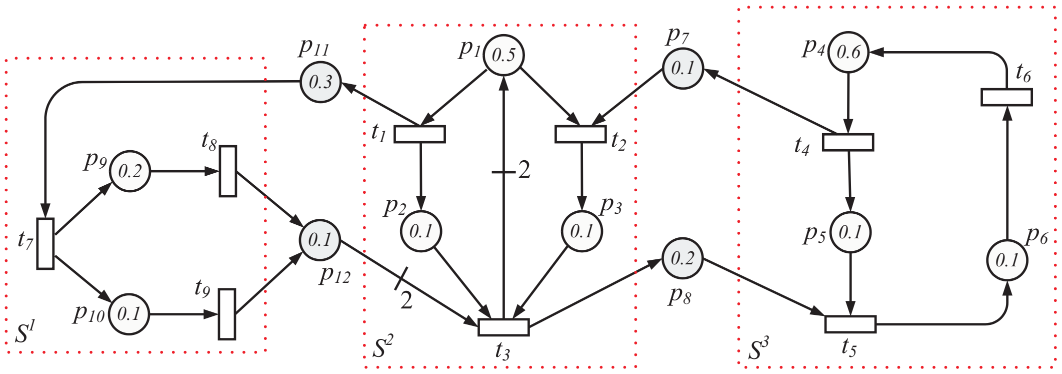


Figure 6.5: A simple TCPN example composed of 3 subsystems

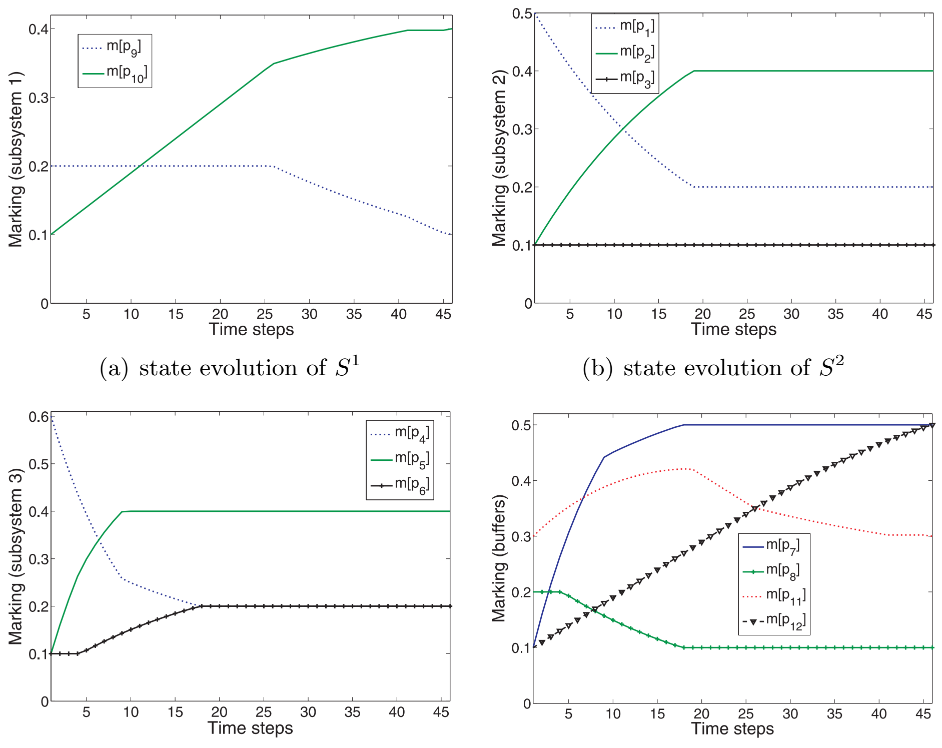
by its neighboring subsytems. Therefore, when a subsystem has reached its final state, it should propagate this information to all the other subsystems, through the connections among neighbors.

6.3.3 A simple example

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

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

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



(a) state evolution of S^1

(b) state evolution of S^2

(c) state evolution of S^3

(d) state evolution of buffers

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

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

6.4 Conclusions

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

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

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

Chapter 7

Minimum-time Flow Control of CF nets

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

7.1 Motivations

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

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

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

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

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

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

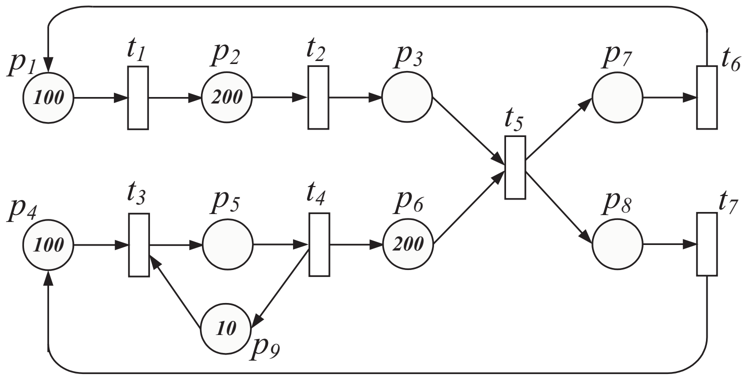
where $\boldsymbol{f}$ is the (uncontrolled) flow vector at marking $\boldsymbol{m}$. Any state in $\mathcal{M}$ is an equilibrium point corresponding to the maximal flow that can be maintained by applying an appropriate control $\boldsymbol{u}$. Because all the constrains of (7.2) are linear, $\mathcal{M}$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

7.2 Difficulties of Minimum-time Flow Control

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

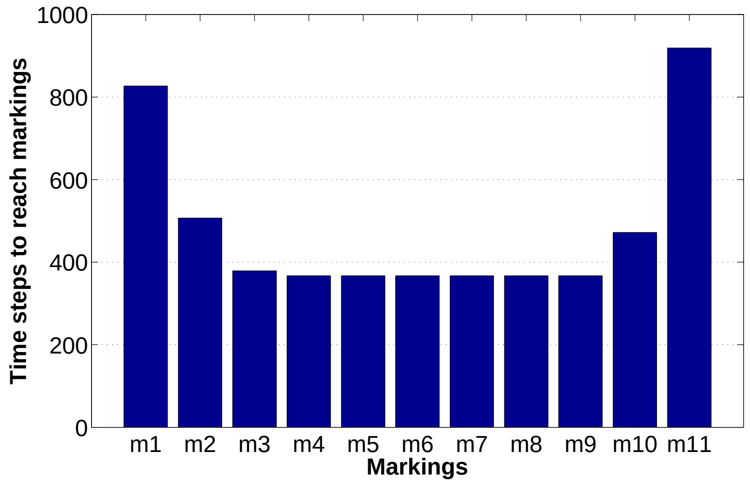
Let us consider the MG (a subclass of CF nets) in Fig.7.1 and assume that sampling period $\Theta = 0.01$. By solving LPP (7.1), we obtain that the maximal flow of any transition (because in MGs $\mathbf{1}$ is the unique T-semiflow) is $\psi = 1$. Two of the possible steady-states (in the same region)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

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



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

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



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

7.3 A heuristic algorithm for CF nets

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1: compute $\psi_j$ by solving LPP (7.1);
2: compute $\boldsymbol{\mu}$ that satisfies (7.3);
3: $k \leftarrow 0$;
4: compute $\boldsymbol{\sigma}_k$ by solving LPP (7.5);
5: while not $(\boldsymbol{m}_k \geq \boldsymbol{\mu})$ do
6:   for all $t \in T$ do
7:     if $\boldsymbol{\sigma}_k[t] > 0$ then
8:       compute the controlled flow $\boldsymbol{w}_k[t]$ corresponding to the ON/OFF strategy;
9:     else if $\boldsymbol{m}_k[p_i] > \boldsymbol{\mu}[p_i]$ for any $p_i \in {}^\bullet t$ then
10:      $\boldsymbol{w}_k[t] \leftarrow \boldsymbol{\lambda}[t] \cdot enab(\boldsymbol{m}_k, t)$;
11:    else
12:      $\boldsymbol{w}_k[t] \leftarrow 0$;
13:    end if
14:  end for
15:  update state: $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \Theta \cdot \boldsymbol{C} \cdot \boldsymbol{w}_k$;
16:  $k \leftarrow k + 1$;
17: end while
18: compute the steady state controlled flow $\boldsymbol{w}_k$, such that: $\boldsymbol{C} \cdot \boldsymbol{w}_k = 0, \boldsymbol{w}_k[t_j] = \psi_j$;
19: Return $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, ..., $\boldsymbol{w}_k$;
```

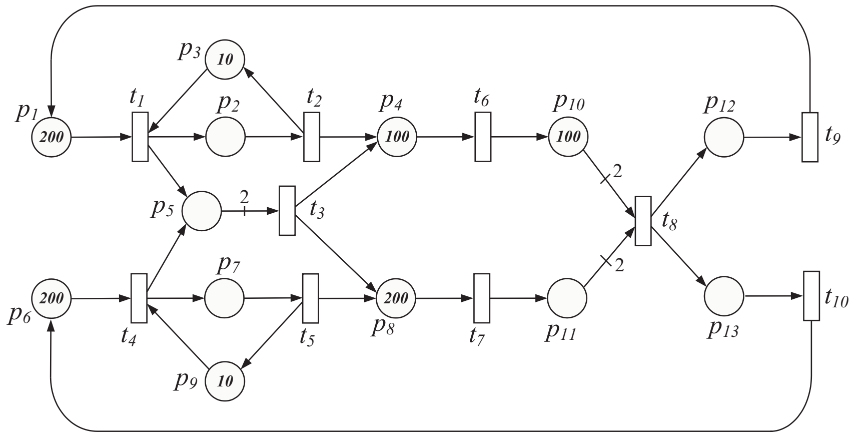
Proof: Since $\mathcal{N}$ is a CF net, the additional firings (for a transition t with $\boldsymbol{\sigma}_k[t] = 0$) do not disable the firing of $\boldsymbol{\sigma}_k$ that drives the system to a state $\boldsymbol{m} \in \mathcal{M}$. On the other hand, for any place p_i with $\boldsymbol{m}[p_i]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

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

7.4 Examples

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

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

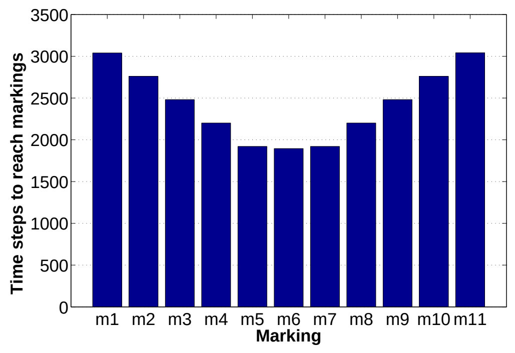


Assume that we want to maximize the flow of transition t_9, by solving LPP (7.1) with $t_j = t_9$, it is obtained $\psi_j = 1$. From (7.3), the corresponding minimal required marking is $\boldsymbol{\mu} = [4\ 6\ 4\ 20\ 40\ 4\ 6\ 4\ 4\ 40\ 40\ 30\ 30]^T$. However, the solution of LPP (7.4) is not unique.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

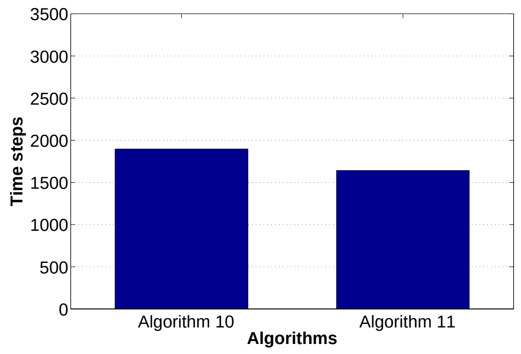
As shown in Fig. 7.4, by applying Algorithm 10 we can obtain the maximal flow in 1895 time steps, which is the same as using the ON/OFF controller to drive the system to $\boldsymbol{m}_6$. However, we should remember that we do not know *a priori* that driving the system to $\boldsymbol{m}_6$ will cost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

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

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



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



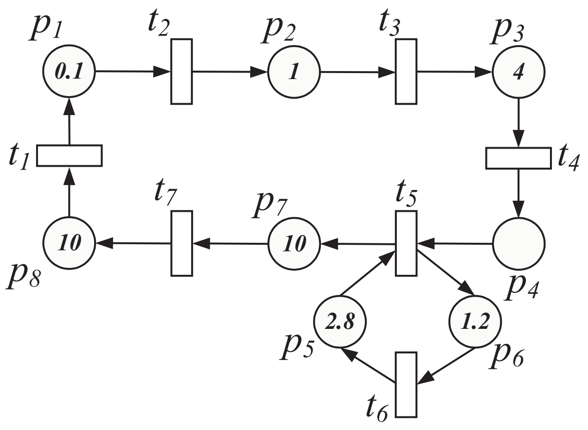
(b) time steps required to reach the maximal flow by applying the proposed algorithms

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

7.5 Conclusions

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

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



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

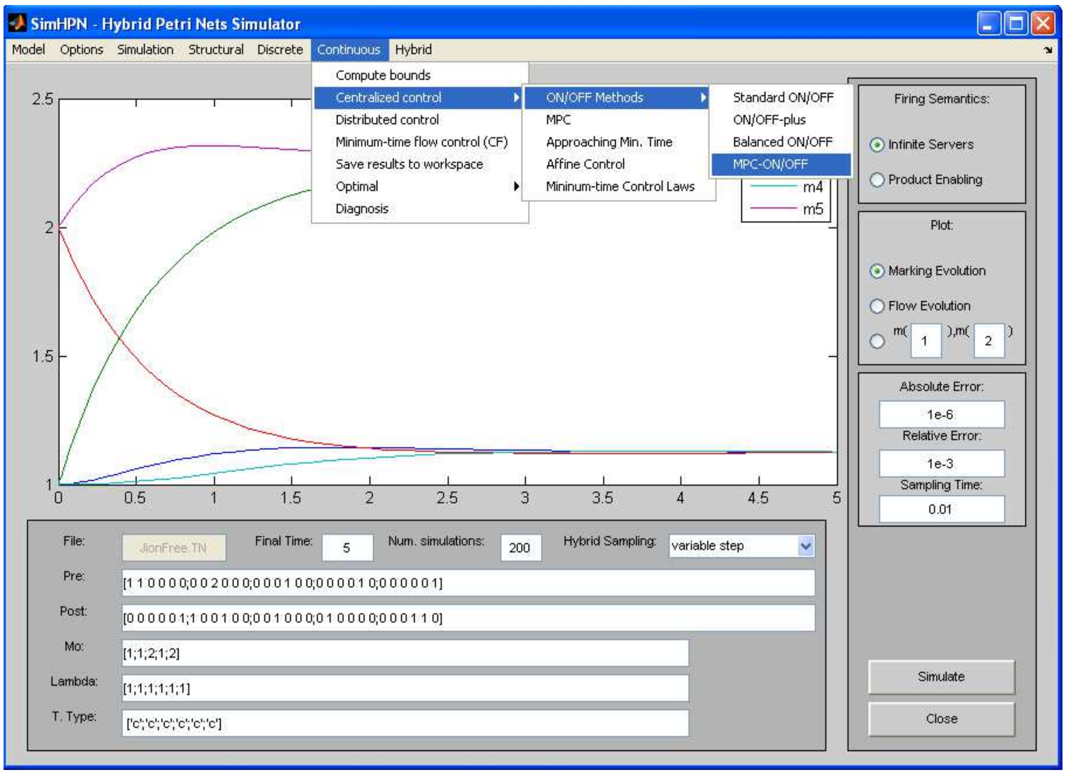
Chapter 8

Simulations and Comparisons

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

8.1 Implementation: SimHPN

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



Firing Semantics: [✓ Infinite Servers] Product Enabling

Plot: [✓ Marking Evolution] Flow Evolution m(1),m(2)

Absolute Error: 1e-6

Relative Error: 1e-3

Sampling Time: 0.01

File: JionFree.TN

Final Time: 5

Num. simulations: 200

Hybrid Sampling: variable step

Pre: [1 1 0 0 0 0;0 0 2 0 0 0;0 0 0 1 0 0;0 0 0 0 1 0;0 0 0 0 0 1]

Post: [0 0 0 0 0 1;1 0 0 1 0 0;0 0 1 0 0 0;0 1 0 0 0 0;0 0 0 1 1 0]

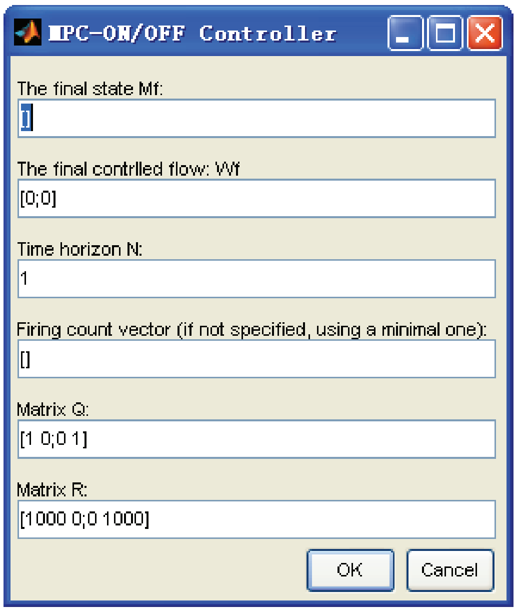
Mo: [1;1;2;1;2]

Lambda: [1;1;1;1;1;1]

T. Type: ['c';'c';'c';'c';'c';'c']

Figure 8.1: The main Graphical User Interface of SimHPN

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


MPC-ON/OFF Controller

The final state Mf:

[]

The final contrlled flow: Wf

[0;0]

Time horizon N:

1

Firing count vector (if not specified, using a minimal one):

[]

Matrix Q:

[1 0;0 1]

Matrix R:

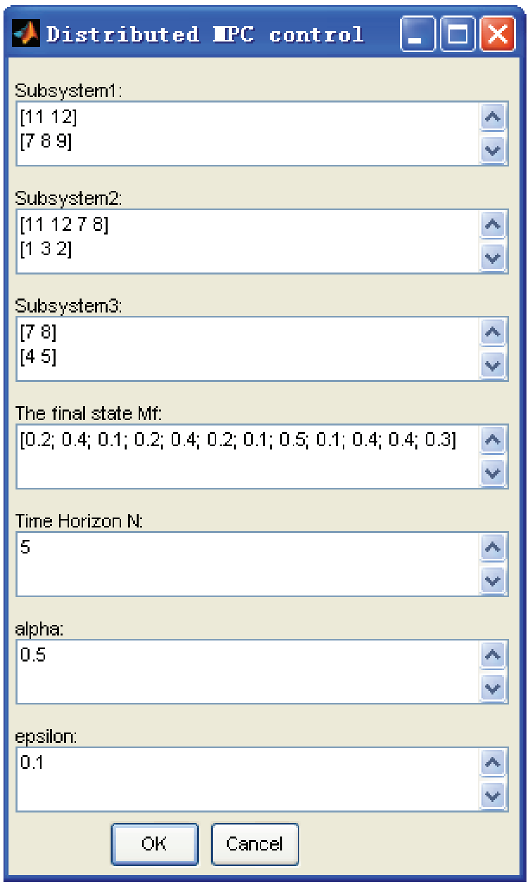
[1000 0;0 1000]

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

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

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

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



Subsystem1:
[11 12]
[7 8 9]

Subsystem2:
[11 12 7 8]
[1 3 2]

Subsystem3:
[7 8]
[4 5]

The final state Mf:
[0.2; 0.4; 0.1; 0.2; 0.4; 0.2; 0.1; 0.5; 0.1; 0.4; 0.4; 0.3]

Time Horizon N:
5

alpha:
0.5

epsilon:
0.1

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

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

8.2 Case study 1: centralized control of a manufacturing cell

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

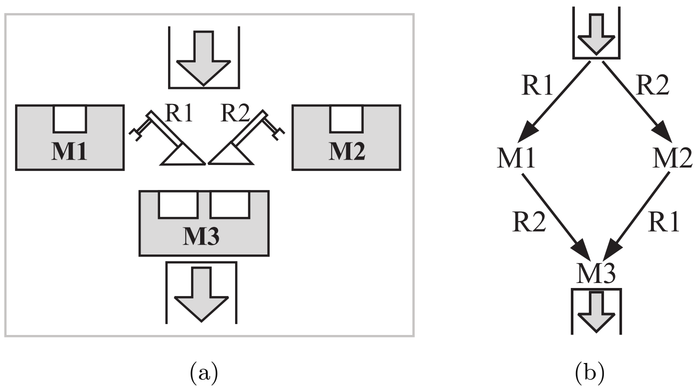


Figure 8.4: (a) Logical layout of a manufacturing cell (b) Its production process

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

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

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

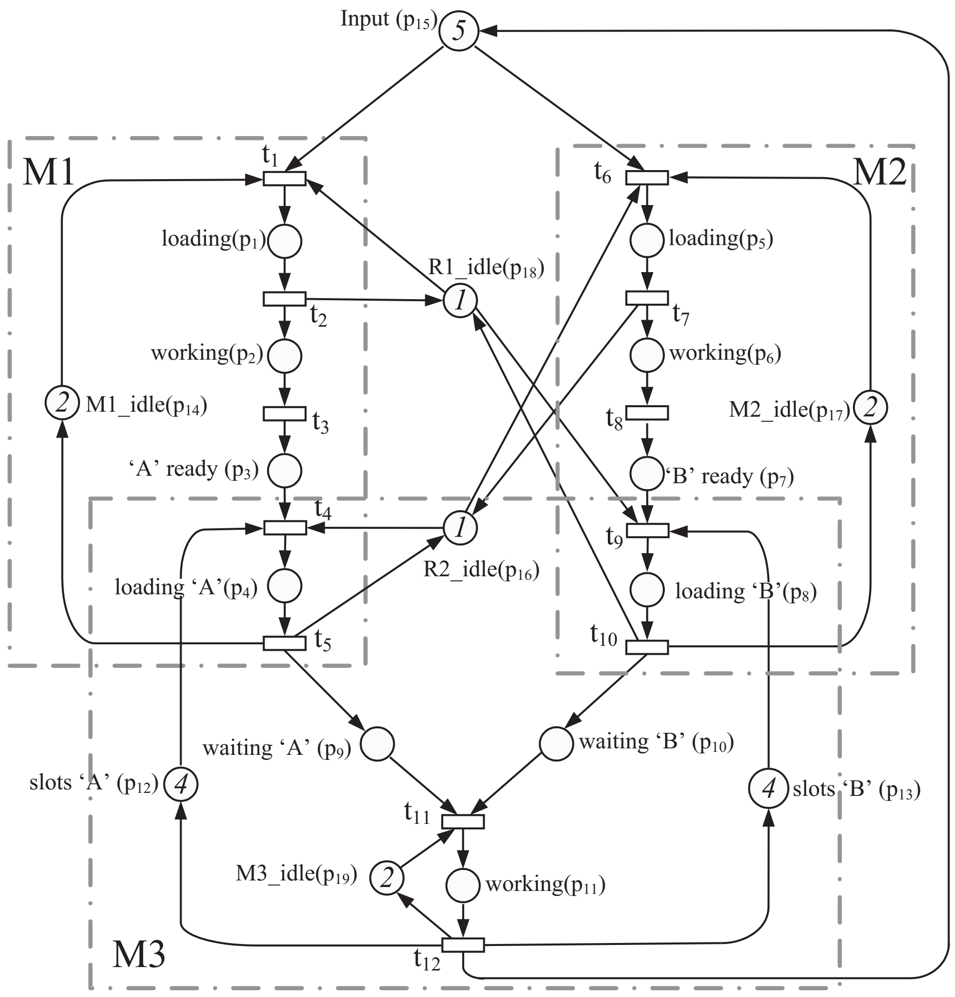


Figure 8.5: The PN model of a manufacturing cell, where robots R1 and R2 are shared resources: R1 is used to move raw materials into M1, and to move semi-products from M2 to M3; R2 is used to move raw materials into M2, and to move semi-products from M1 to M3.

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

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

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

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

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

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

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

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

We also consider a variant of the net system in Fig. 8.5, shown in Fig. 8.6. The difference is that now the robots R1, R2 are used in a different manner: R1 is shared by machine M1 and M2 to move raw materials into machines; while R2 is shared by machine M1 and M2 to move semi-products into M3. The same initial marking and firing rates are used as before and by solving the same LPPs, we obtain the maximal flow at a final marking with the minimal WIP cost: $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.58\ 1.083\ 1.045\ 0.55]^T$.

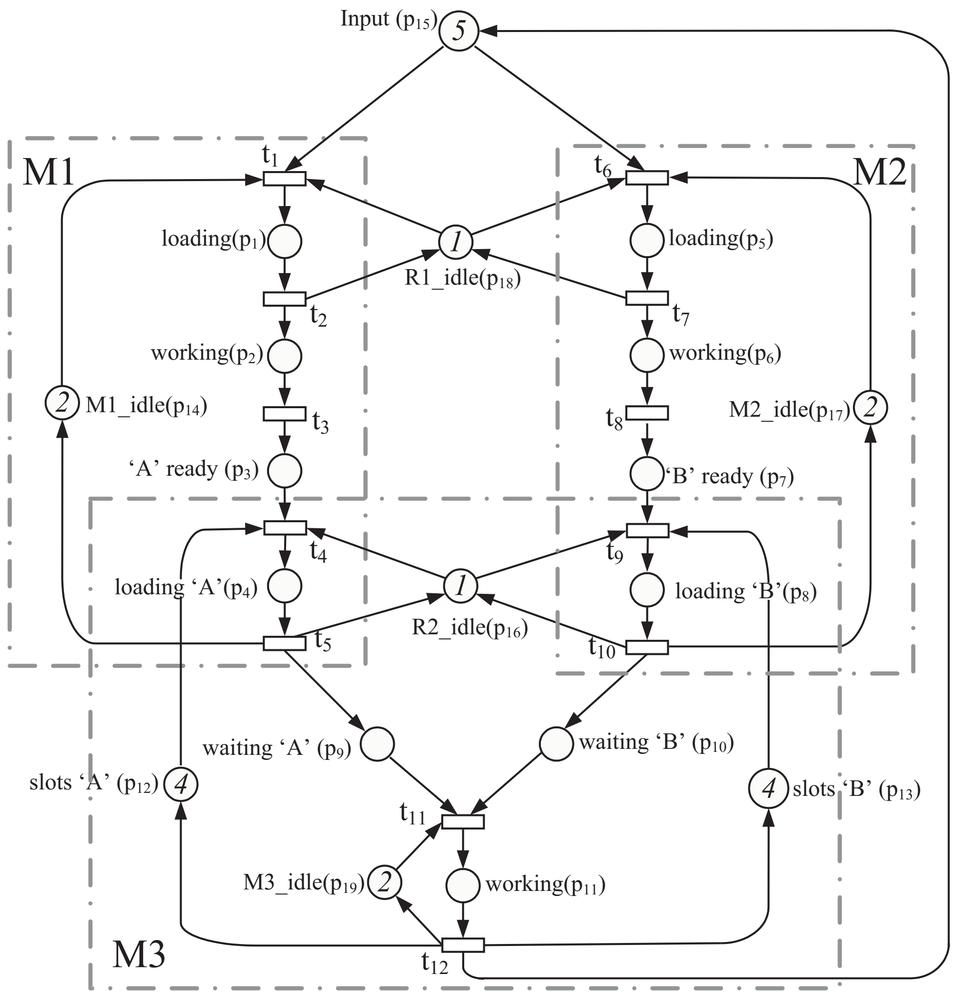


Figure 8.6: A variant of the PN model in Fig. 8.5: Robots R1, R2 are used in a different manner. Now R1 is used to move raw materials to M1 and M2, and R2 is used to move semi-products from M1 or M2 to M3.

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

For the original system in Fig. 8.5, the B-ON/OFF controller gives the smallest number of time steps (209), obtained by using $d = 2$. The ON/OFF+ controller requires the largest number of time steps (457), although its computational cost (275ms) is only about half of the B-ON/OFF controller. The result of the approaching minimum-time controller is not very good, it requires 311 time steps to reach the final state and the required CPU time to compute the control law is 200 times as large as that of the B-ON/OFF controller. The number of time steps of the MPC-ON/OFF controller (around 220) is not far from the best, however its computational cost is high, for example, when $N = 5$ it is more than 200 times higher than that of the B-ON/OFF controller.

For the variant system in Fig. 8.6, the B-ON/OFF controller still gives the smallest number of time steps (223). The same number is also obtained by using the ON/OFF+ controller, while its computational cost is also the lowest (108ms, about one fourth of that of the B-ON/OFF controller). The approaching minimum-time controller requires the largest number of time steps (316) and very high computational cost (55,608ms). The MPC-ON/OFF controller also gives quite small number of time steps (228) but requires much more CPU time (even with $N = 1$ it is about 40 times larger than that of the ON/OFF+ controller.)

8.3 Case study 2: centralized control of an assembly line

The second example (adapted from [111]) is a more complex assembly line with five machines, three different parts A, B and C are assembled for one final product. The input of part A is first processed in machine M1 then machine M3; the input of part B is processed by machine M2 then machine M1; the semi-products from part A and B are first processed in machine M4; finally the obtained products are assembly in machine M5 with the one from part C that is sequentially processed by machine M3, M1 and M4. The production process of the system is shown in Fig. 8.7.

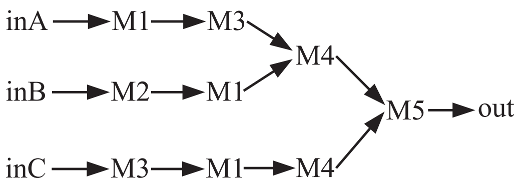


Figure 8.7: The production process of an assembly system

The PN model of the system is presented in Fig. 8.8. Machines are modelled by resource places labelled with name M1 to M5; each machine has buffers to store the semi-products, for instance, after the input of part A has been processed by M1, the obtained products are stored in buffer place $B1A(p_3)$, and the maximal sizes of the buffers are limited by, for example, place $MaxB1A(p_{23})$. The customer orders for the final products are also modelled and a "push" strategy is applied, i.e., the assembling process keeps working until the buffers are full. This strategy can decrease the customer waiting time, accepting a high work in process. The interpretations of the places and transitions in the PN model are explained in Table 8.2.

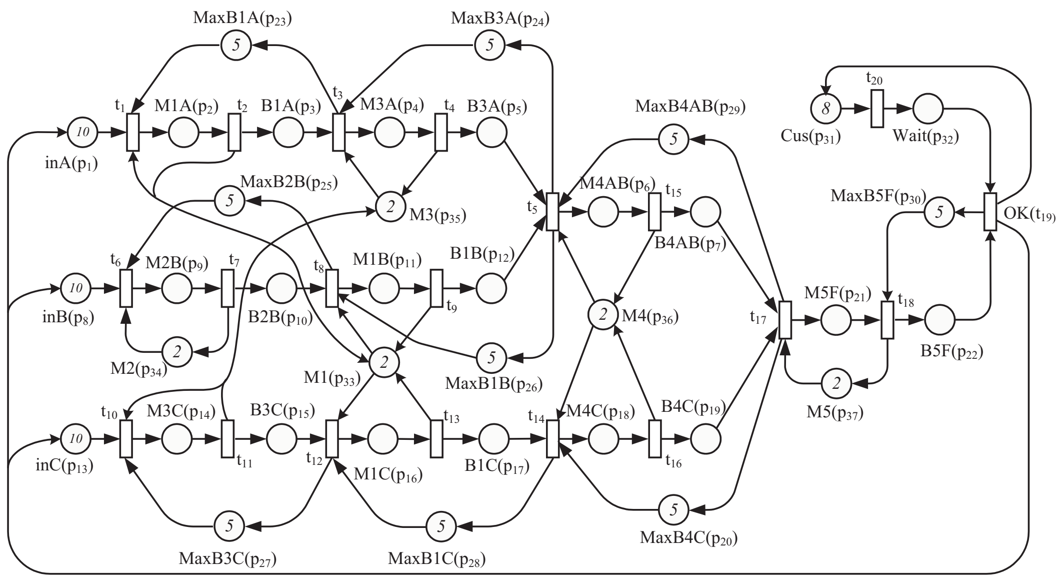


Figure 8.8: The PN model of an assembly system with five machines

Table 8.2: The interpretation of the PN model in Fig. 8.8

Place	Interpretation	Transition	Interpretation
inX	input of part X	t_1, t_8, t_{12}	M1 starts working
My	machine y	t_2, t_9, t_{13}	M1 finises
MyX	machine y is working on part X	t_6	M2 starts working
M5F	machine 5 is working on the final product	t_7	M2 finishes
ByX	buffer of part X in My	t_3, t_{10}	M3 starts working
B5F	buffer of the final product in M5	t_4, t_{11}	M3 finishes
MaxByX	maximal size of buffer ByX	t_5, t_{14}	M4 starts working
MaxB5F	maximal size of buffer B5F	t_{15}, t_{16}	M4 finishes
Cus	customers	t_{17}	M5 starts working
Wait	waiting orders	t_{18}	M3 finishes
		t_{19}	final product delivers
		t_{20}	customer order arrives

⋆ X = A, B, C and y = 1, 2, 3, 4.

We assume that the input of each part initially has 10 pieces; each machine can handle 2 pieces at the same time; buffer sizes are limited to 5; and initially 8 customers are considered. The corresponding initial marking $\boldsymbol{m}_0$ is shown in Fig. 8.8, in which the empty places are assumed to have initial markings equal to 0.1. Similarly to the first example, we also assume that every transition has an average delay time, denoted by $\boldsymbol{\delta}$: the stating time delays of all machines are equal to 0.02 time units, i.e., $\boldsymbol{\delta}[t_1] = \boldsymbol{\delta}[t_3] = \boldsymbol{\delta}[t_5] = \boldsymbol{\delta}[t_6] = \boldsymbol{\delta}[t_8] = \boldsymbol{\delta}[t_{10}] = \boldsymbol{\delta}[t_{12}] = \boldsymbol{\delta}[t_{14}] = \boldsymbol{\delta}[t_{17}] = 0.02$; the working process of machine M1 takes 0.2 time units, i.e., $\boldsymbol{\delta}[t_2] = \boldsymbol{\delta}[t_9] = \boldsymbol{\delta}[t_{13}] = 0.2$; for machines M3 and M4, 0.4 time units, i.e., $\boldsymbol{\delta}[t_4] = \boldsymbol{\delta}[t_{11}] = \boldsymbol{\delta}[t_{15}] = \boldsymbol{\delta}[t_{16}] = 0.4$; machines M2, M5 work slower, it takes 0.8 time units, i.e., $\boldsymbol{\delta}[t_{7]} = \boldsymbol{\delta}[t_{18}] = 0.8$; the delay time of customer orders is 0.8; and the final products are delivered with delay of 0.1 time units, i.e., $\boldsymbol{\delta}[t_{19}] = 0.1$.

We consider reaching a final state with the flow of transition t_{19} (modelling the delivering of final products) maximized, and now we will also consider the work in process (WIP) cost in the profit function. We assume a fixed residence cost, equal to 50, for per piece of (semi-) products in the buffers; for machines M1, M4, the operation cost is 150; for machines M2, M5, the cost is 100; and for machine M3, the cost is 120. For per piece of final products, we assume an income of 1000. Then the following LPP can be written:

$$\begin{array}{ll} \max & J = 1000 \cdot \boldsymbol{w}_f[t_{19}] - \boldsymbol{l} \cdot \boldsymbol{m}_f \\ \text{s.t.} & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_f = \boldsymbol{0} \\ & \boldsymbol{w}_f[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad\qquad \forall p_i \in {}^{\bullet}t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_f, \boldsymbol{\sigma}, \boldsymbol{m}_f \geq \boldsymbol{0} \end{array} \tag{8.2}$$

where J is the profit function and $\boldsymbol{l}[3] = \boldsymbol{l}[5] = \boldsymbol{l}[10] = \boldsymbol{l}[12] = \boldsymbol{l}[15] = \boldsymbol{l}[17] = \boldsymbol{l}[7] = \boldsymbol{l}[19] = \boldsymbol{l}[22] = 50$, $\boldsymbol{l}[2] = \boldsymbol{l}[11] = \boldsymbol{l}[16] = \boldsymbol{l}[6] = \boldsymbol{l}[18] = 150$, $\boldsymbol{l}[4] = \boldsymbol{l}[14] = 120$, $\boldsymbol{l}[9] = \boldsymbol{l}[21] = 100$, for other places p_i, $\boldsymbol{l}[i] = 0$.

By solving LPP (8.2), a desired final state is $\boldsymbol{m}_f$ = [5.78 0.5122 0.05122 1.024 0.05122 1.024 0.05122 4.756 2.049 0.05122 0.5122 0.05122 5.78 1.024 0.05122 0.5122 0.05122 1.024 0.05122 4.124 2.049 0.2561 4.637 4.124 3.1 4.637 4.124 4.637 4.124 4.844 7.844 0.2561 0.7634 0.05122 0.1512 0.1512 0.05122]^T, with maximal flow $\boldsymbol{w}_f[t_{19}] = 2.561$. Table A.7 shows the simulation results by using different control methods.

In this case, the B-ON/OFF controller again gives the smallest number of time steps (131), which is much smaller than that of the ON/OFF+ controller (249) and that of the approaching minimum-time controller (300). By using the MPC-ON/OFF controller with $N = 5$, the same number of time steps as that of the B-ON/OFF controller can be obtained, but its computational cost (284,859ms) is about 600 times larger than that of the B-ON/OFF.

8.4 Case study 3: centralized control of a manufacturing system

In the third example (taken from [60]), we consider a flexible manufacturing system with four types of machines M1 to M4, and three type of robots R1 to R3. Three type of products P1 to P3 can be produced in this system. Fig. 8.9 represents its production process. The PN model of this system is shown in Fig.8.10, and the final products are finished in transition t_{14}, t_{20}, t_6 respectively.

$$\begin{tikzcd}
\text{P1 : I1} \arrow[r, "\text{R2}"] & \text{M2} \arrow[r, "\text{R2}"] & \text{O1} & \\
\text{P2 : I2} \arrow[r, "\text{R3}"] & \text{M4} \arrow[r, "\text{R2}"] & \text{M3} \arrow[r, "\text{R1}"] & \text{O2} \\
& \text{M3} \arrow[r, "\text{R2}"] & \text{M4} \arrow[dr, "\text{R3}"] & \\
\text{P3 : I3} \arrow[ur, "\text{R1}"] \arrow[dr, "\text{R1}"'] & & & \text{O3} \\
& \text{M1} \arrow[r, "\text{R2}"] & \text{M2} \arrow[ur, "\text{R3}"'] &
\end{tikzcd}$$

Figure 8.9: The production process of a flexible manufacturing system

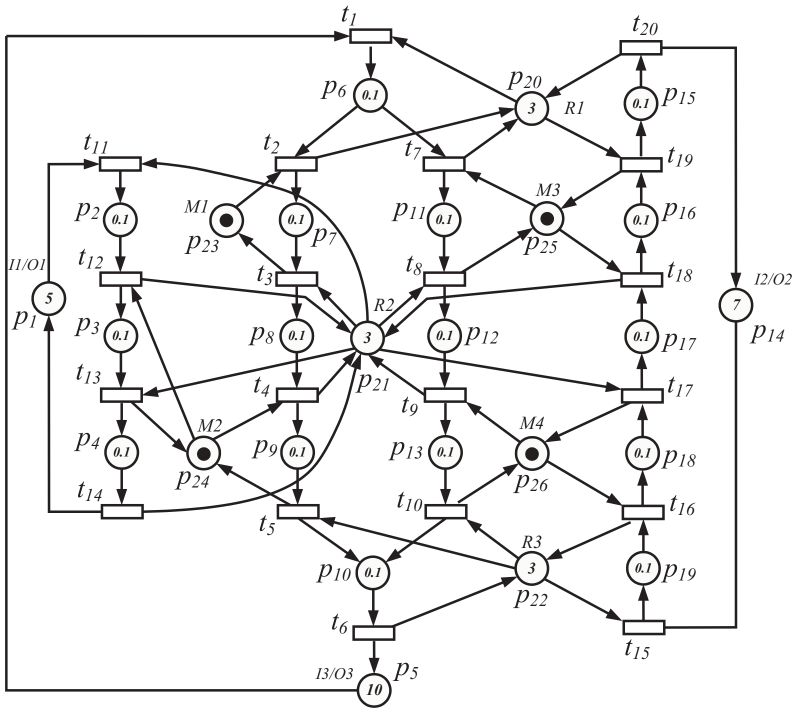


Figure 8.10: The PN model of a flexible manufacturing system

Assume that the initial state $\boldsymbol{m}_0(p_1) = 5$, $\boldsymbol{m}_0(p_5) = 10$, $\boldsymbol{m}_0(p_{14}) = 7$, $\boldsymbol{m}_0(p_{20}) =$

$\boldsymbol{m}_0(p_{21}) = \boldsymbol{m}_0(p_{22}) = 3$, $\boldsymbol{m}_0(p_{23}) = \boldsymbol{m}_0(p_{24}) = \boldsymbol{m}_0(p_{25}) = \boldsymbol{m}_0(p_{26}) = 1$ and the initial marking of the rest of places is equal to 0.1, the firing rate of transitions is defined as $\boldsymbol{\lambda}$ =[1/3 1/2 1/2 1/2 1/2 1 1/2 1/2 1/2 1/2 1/4 1/2 1/2 1 1/2 1/2 1/2 1/2 1/2 1/4]^T and sampling period be $\Theta = 0.03$. Let us assume that the objective function of the system is defined as: $3 \cdot \boldsymbol{w}_f[t_6] + 5 \cdot \boldsymbol{w}_f[t_{14}] + 4 \cdot \boldsymbol{w}_f[t_{20}]$, where $\boldsymbol{w}_f$ is the flow in a steady state. Then following LPP can be written:

$$\begin{array}{ll} \max & J = 3 \cdot \boldsymbol{w}_f[t_6] + 5 \cdot \boldsymbol{w}_f[t_{14}] + 4 \cdot \boldsymbol{w}_f[t_{20}] \\ \text{s.t.} & \boldsymbol{m}_f = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\ & \boldsymbol{C} \cdot \boldsymbol{w}_f = \boldsymbol{0} \\ & \boldsymbol{w}_f[t] = \boldsymbol{\lambda}[t] \cdot \frac{\boldsymbol{m}_f[p_i]}{\boldsymbol{Pre}[p_i,t]} - \boldsymbol{v}[p_i,t], \\ & \qquad\qquad \forall p_i \in {}^{\bullet}t, \boldsymbol{v}[p_i,t] \geq 0 \\ & \boldsymbol{w}_f, \boldsymbol{\sigma}, \boldsymbol{m}_f \geq \boldsymbol{0} \end{array} \tag{8.3}$$

By solving LPP (8.3), a desired final state (maximizing the profit function) can be computed: $\boldsymbol{m}_f$ = [3.4; 1.3; 0.4; 0.2; 7.6; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 0.4; 3.1; 0.8; 0.4; 0.4; 0.4; 2.4; 2; 0.8; 0.4; 0.7; 0.4; 0.4; 0.4] and the maximal profit is $J = 3$. The simulation results by using different control methods are shown in Table A.8.

In this example, the approaching minimum-time controller gives the smallest number of time steps (279), which is about 10% smaller than that of the other control methods. However, the CPU time required for computing the control law (60,691ms) is about 200 times as large as that of the ON/OFF+ controller and 50 times as large as that of the B-ON/OFF controller.

8.5 Discussions of case study 1–3

In Table 8.3, we summarize the smallest numbers of time steps that obtained by using different control methods, and the corresponding parameters, according to the simulation results shown in Table A.5–A.8. From the simulation results, we can make some initial conclusions (that are consistent with those in Section 4.4.5):

Regarding the numbers of time steps:

- The B-ON/OFF controller gives the smallest number of time steps in most of the cases (except for case study 3). Usually smaller numbers of time steps are obtained by using smaller values of d, as shown in Table A.5 and Table A.7. A small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows, therefore a smaller value of d may give better results. If d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to apply the ON/OFF+ directly. However, in the case that the B-ON/OFF controller cannot improve the result of the ON/OFF+ controller

Table 8.3: The smallest numbers of time steps of using different control methods among different parameters (if exist), derived from Table A.5–A.8. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Cases	Control methods	Time steps	CPU time	Parameters
Case 1	appro. min-time	311	88,505	
	ON/OFF+	457	275	
	B-ON/OFF	209	426	$d = 2$
	MPC-ON/OFF	219	102,622	$N = 5$, $r = 1000$, $q = 1$ (or 100, 1000)
Case 1 (variant)	appro. min-time	316	55,608	
	ON/OFF+	223	108	
	B-ON/OFF	223	419	$d = 20$ (or 15, 10, 5, 2)
	MPC-ON/OFF	228	5,018	$N = 1$, r=1000, q=1
	MPC-ON/OFF	228	99,749	$N = 5$, $r = 1000$, $q = 1$ (or 100, 1000)
Case 2	appro. min-time	300	138,826	
	ON/OFF+	249	213	
	B-ON/OFF	131	475	$d = 10$ (or 5, 2, 1)
	MPC-ON/OFF	131	284,859	$N = 5$, r=1000, q=100
Case 3	appro. min-time	279	60,691	
	ON/OFF+	301	271	
	B-ON/OFF	301	1,157	$d = 20$ (or 15, 10, 5, 2, 1)
	MPC-ON/OFF	310	443,439	$N = 5$, $r = 1000$, $q = 1$ (or 100, 1000)

(when conflicting transitions have similar flows), the numbers of time steps are not sensitive to the values of d, as shown in Table A.6 and Table A.8.

- The ON/OFF+ controller usually gives quite small numbers of time steps (as shown in Table A.6 and Table A.8), except when the flows of conflicting transitions are very different. For example, in the original system in case study 1 (Fig. 8.5, Table A.5), the flows of t_4 and t_9 are much smaller than the ones of t_1 and t_6; those four transitions are in a coupled conflict, therefore if we fire them proportionally by using the ON/OFF+ strategy the result is not good, costing 457 time steps, more than the double of the B-ON/OFF controller, to reach $\boldsymbol{m}_f$. On the other hand, we can see that in the variant system in cast study 1 (Fig. 8.6, Table A.6), it gives the smallest number of time steps (as the B-ON/OFF controller) and its computational cost is the lowest. It is because the fact that in the variant system, transitions t_4, t_9 and transitions t_1, t_6 are now in different sets of coupled conflict.

- Most probably, the approaching minimum-time controller may not work very well when there exist some places with very small initial markings (as in case study 1 and 2, Table A.5–A.7). The reason may be what we have mentioned before: in this approach the flow between two adjacent states is determined by the one with the smaller flow, so if some states have very small flows (for example the initial state), it may take long time to reach the final state.

- The numbers of time steps obtained by using the MPC-ON/OFF controller are not far from the best among other control methods, even the best (as the B-ON/OFF controller) in case 2 (shown in Table A.7). Usually, its required numbers of time steps can be decreased but using a larger time horizon N, but the improvement is not very significant (less than 0.5% in case 1, 3% in case 3 and 10% in case 2, but the computational costs increase more than 20 times). However, we can also observe that even with a small N, we may already obtain a reasonable number of time steps (as in case 1, Table A.5 and Table A.6). On the other hands, in these examples the number of time steps is not very sensitive to the weights on the matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Regarding the computational costs:

- The ON/OFF+ controller is the computationally cheapest one in all these three case studies.
- The B-ON/OFF controller costs slightly higher CPU time than the ON/OFF+ controller (less than 5 times higher), because an estimation of number of time steps should be computed at each time step. But, it is still very efficient.
- We can see that in those three cases, the approaching minimum-time controller costs more than hundred times the CPU time than that of the ON/OFF+ and B-ON/OFF controller, because it needs to solve a non-linear problem whenever an intermediate state is inserted on the trajectory to reduce the time.
- In these examples, the computational cost of MPC-ON/OFF controller is not higher than the approaching minimum-time controller when a small N is used; but it always costs more CPU time than the ON/OFF+ and B-ON/OFF controller. On the other hand, the computational cost grows fast with respect to N. From our simulation, the required CPU time for computing the control law increases more than 20 times, when N increases from 1 to 5. (We should also notice that, the increasing speed of the computational costs with respect to N may also depend on the size and structure of the considered net systems.)

8.6 Case study 4: distributed control of an AGV System

In this case study we consider a model of Automatic Guided Vehicle Systems(AGVS). The system describes a manufacturing factory floor that consists of four workstations: WS 1 to 4 and three AGVS areas: AGVS 1 to 3. Each workstation handles some parts of the system that are first stored in the buffers, then moved from one workstation to another through the AGVS areas. The system can be naturally viewed as 7 subsystems connected by those buffers, each workstation or AGVS area as a subsystem. The PN model of the system is presented in Fig. 8.11, in which the buffers are modelled as places. The input and output buffer places of each subsystem are: for subsystem S^1, $B^{(\cdot,1)} = \{p_1\}$, $B^{(1,\cdot)} = \{p_9, p_{10}\}$; for subsystem S^2, $B^{(\cdot,2)} = \{p_9, p_{10}\}$, $B^{(2,\cdot)} = \{p_{16}, p_{24}\}$; for subsystem S^3, $B^{(\cdot,3)} = \{p_{16}\}$, $B^{(3,\cdot)} = \{p_{30}\}$; for subsystem S^4, $B^{(\cdot,4)} = \{p_{24}\}$, $B^{(4,\cdot)} = \{p_{27}\}$; for subsystem S^5, $B^{(\cdot,5)} = \{p_{30}, p_{27}\}$, $B^{(5,\cdot)} = \{p_{33}, p_{36}\}$; for subsystem S^6, $B^{(\cdot,6)} = \{p_{33}, p_{36}\}$, $B^{(6,\cdot)} = \{p_{45}\}$; for subsystem S^7, $B^{(\cdot,7)} = \{p_{45}\}$, $B^{(7,\cdot)} = \{p_1\}$.

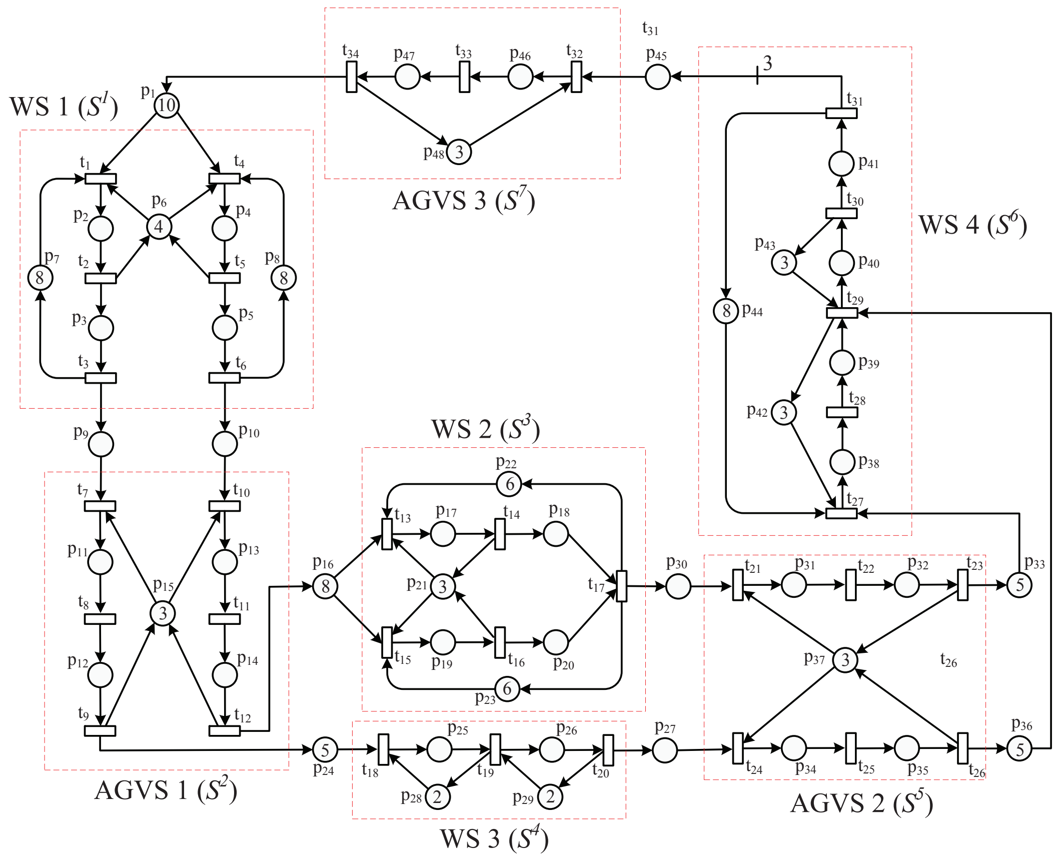


Figure 8.11: The PN model of an AGVS composed by 7 subsystems

We assume that in the initial state there are certain raw materials or parts in the input buffer places of each workstation, in particular, $\boldsymbol{m}_0[p_1] = 10$; $\boldsymbol{m}_0[p_{16}] = 8$; $\boldsymbol{m}_0[p_{24}] = 5$; $\boldsymbol{m}_0[p_{33}] = \boldsymbol{m}_0[p_{36}] = 5$. In each workstation there exist some (shared) machines, modelled by places, which have the initial markings as the following: $\boldsymbol{m}_0[p_6] = 4$; $\boldsymbol{m}_0[p_{21}] = 3$; $\boldsymbol{m}_0[p_{28}] = \boldsymbol{m}_0[p_{29}] = 2$; $\boldsymbol{m}_0[p_{42}] = \boldsymbol{m}_0[p_{43}] = 3$. There also exist limitations for the parts that can be accepted by the workstations: $\boldsymbol{m}_0[p_7] = \boldsymbol{m}_0[p_8] = 8$; $\boldsymbol{m}_0[p_{22}] = \boldsymbol{m}_0[p_{23}] = 6$; $\boldsymbol{m}_0[p_{44}] = 8$. We assume that in each AGVS area, there maximally have 3 available vehicles, i.e.,

$\boldsymbol{m}_0[p_{15}] = \boldsymbol{m}_0[p_{37}] = \boldsymbol{m}_0[p_{48}] = 3$. For the other places in model, we assume their initial markings equal to 0.1 ensuring positiveness of the initial state required by the control method. Similarly to the previous examples, we assume final states of subsystems given in Table 8.4 such that the maximal flow may be obtained. The final states of buffer places are: $m_f[p_1] = 4.23$; $m_f[p_9] = 0.14$; $m_f[p_{10}] = 0.27$; $m_f[p_{16}] = 4.80$; $m_f[p_{24}] = 3.60$; $m_f[p_{27}] = 0.14$; $m_f[p_{30}] = 0.14$; $m_f[p_{33}] = 1.57$; $m_f[p_{36}] = 3.00$; $m_f[p_{45}] = 0.41$. We assume that the average delay times of transitions are set to the values shown in Table 8.5 and the sampling period $\Theta = 0.05$.

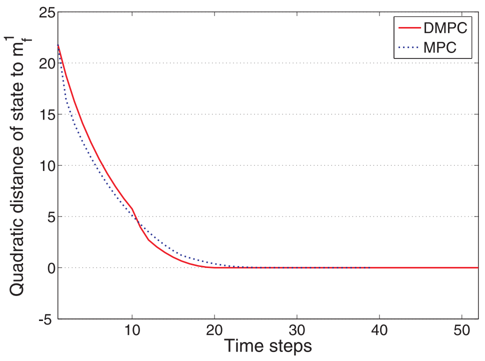
Table 8.4: The final states of subsystems

S^1		S^2		S^3		S^4		S^5		S^6		S^7	
p_2	0.68	p_{11}	0.40	p_{17}	0.54	p_{25}	0.68	p_{31}	0.40	p_{38}	1.08	p_{46}	1.22
p_3	0.54	p_{12}	0.14	p_{18}	0.81	p_{26}	0.54	p_{32}	0.14	p_{39}	0.54	p_{47}	0.40
p_4	1.36	p_{13}	0.81	p_{19}	0.68	p_{28}	1.42	p_{34}	0.40	p_{40}	1.08	p_{48}	1.57
p_5	2.17	p_{14}	0.27	p_{20}	0.81	p_{29}	1.56	p_{35}	0.14	p_{41}	0.54		
p_6	2.17			p_{21}	1.98			p_{37}	2.32	p_{42}	1.57		
p_7	6.98			p_{22}	4.85					p_{43}	2.02		
p_8	4.68			p_{23}	4.71					p_{44}	5.15		

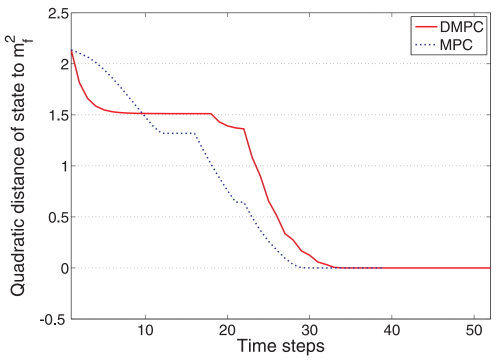
Table 8.5: The average delay times of transitions

S^1		S^2		S^3		S^4		S^5		S^6		S^7	
t_1	0.4	t_7	0.1	t_{13}	0.3	t_{18}	0.4	t_{21}	0.1	t_{27}	0.4	t_{32}	0.1
t_2	0.5	t_8	0.3	t_{14}	0.4	t_{19}	0.5	t_{22}	0.3	t_{28}	0.8	t_{33}	0.3
t_3	0.4	t_9	0.1	t_{15}	0.5	t_{20}	0.4	t_{23}	0.1	t_{29}	0.4	t_{34}	0.1
t_4	0.8	t_{10}	0.1	t_{16}	0.5			t_{24}	0.1	t_{30}	0.8		
t_5	0.5	t_{11}	0.3	t_{17}	0.6			t_{25}	0.3	t_{31}	0.4		
t_6	0.8	t_{12}	0.1					t_{26}	0.1				

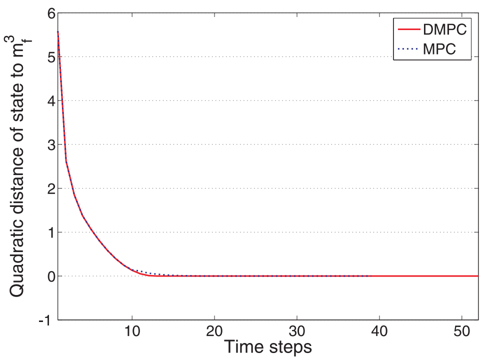
The problem we handle here is to drive all the subsystems to their final states by using distributed control methods. Clearly, the net is not CF, so the approach proposed in Chapter 5 is not applicable; on the other hand, subsystems are not mono-T-semiflow (for example, subsystem S^1 has two minimal T-semiflows), so we cannot apply the method proposed in [4] either. Hence, here we will apply the distributed MPC controller presented in Chapter 6. In Fig. 8.12 and Fig. 8.13, for each subsystem S^l we show the *quadratic distance* of states $\boldsymbol{m}_k^l$ (at any time step k) to the final state $\boldsymbol{m}_f{}^l$, defined as $(\boldsymbol{m}_k^l - \boldsymbol{m}_f{}^l)^T \cdot (\boldsymbol{m}_k^l - \boldsymbol{m}_f{}^l)$, $l = 1$ to 7, by using the centralized MPC controller given in Algorithm 8 and the distributed MPC controller given in Algorithm 9. Since subsystems may not reach their final states at the same time, Fig. 8.13(d) shows the different time instants of reaching the final states by using the centralized and distributed MPC. The results are obtained by using $N = 3$, $\boldsymbol{Q} = \boldsymbol{I}$, $\boldsymbol{Z} = 1000 \cdot \boldsymbol{I}$. For other parameters in distributed MPC, we use $\alpha = 0.5$, $\epsilon_1 = 0.1$, $\epsilon_2 = 10^{-6}$.



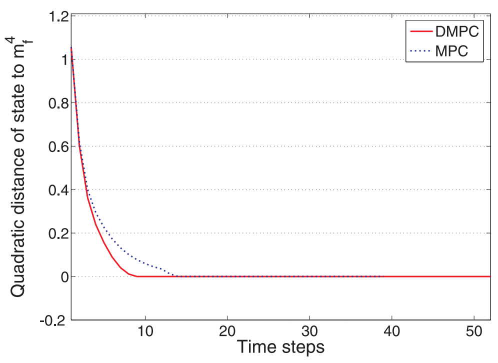
(a) S^1



(b) S^2



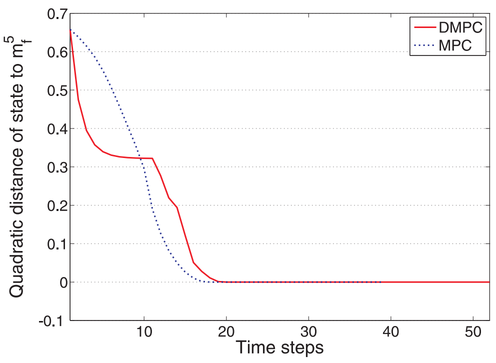
(c) S^3



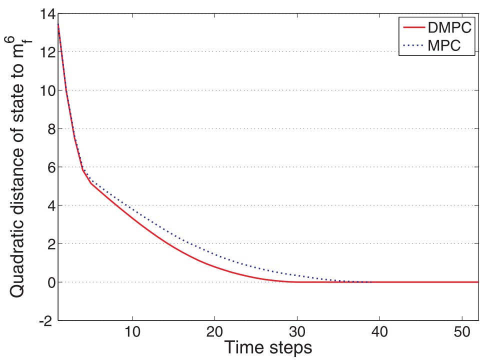
(d) S^4

Figure 8.12: The quadratic distance of $\boldsymbol{m}_k^l$ to the final state $\boldsymbol{m}_f{}^l$, $l = 1, 2, 3, 4$

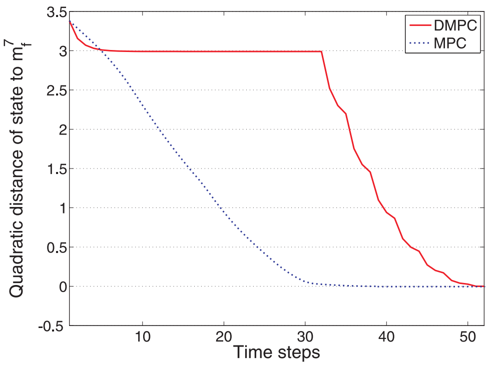
It can be observed that all the subsystems reach their final states in finite time by using both the centralized and distributed methods. From Fig. 8.13(d) we can see that, the last subsystem that reaches its final state is S^7, requiring 38 time steps by using the centralized MPC and 51 time steps by using the distributed MPC. However, let us notice that not all the subsystems reach their final states slower by using the distributed MPC. For instance, the distributed MPC controller requires 9 time steps for subsystem S^4 to reach $\boldsymbol{m}_f{}^4$, smaller than the 14 time steps of using the centralized MPC. On the other hand, Fig. 8.15(a) shows that by using the distributed MPC, the CPU time consumed for computing the control laws (the sum of the CPU time consumed in all the subsystems) is much smaller than by using the centralized MPC, and the reason is clear: the computational complexity of MPC based approaches may grow very quickly on the size of net systems. We should point out that, in both centralized and distributed MPC approaches, from each time step k the state evolutions of subsystems are constrained to be inside the subset convex $R(\mathcal{N}, \boldsymbol{m}_k, \boldsymbol{m}_f)$ (generated by constrains (6.2d) and (6.2e)) of the reachability space.



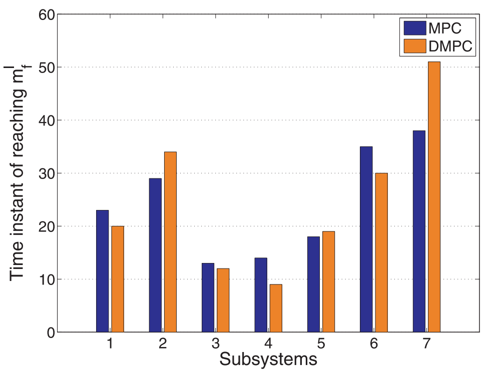
(a) S^5



(b) S^6



(c) S^7

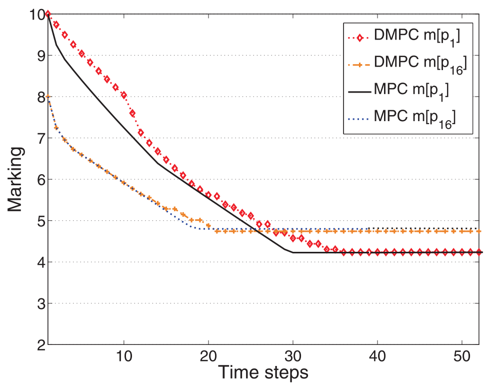


(d) Time instants

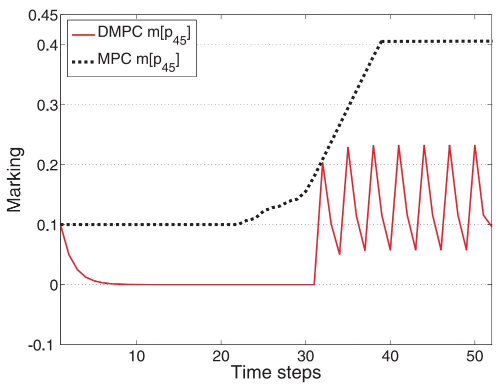
Figure 8.13: (a)-(c) the quadratic distance of $\boldsymbol{m}_k^l$ to the final state $\boldsymbol{m_f}^l$, $l = 5, 6, 7$; (d) the time instants of subsystems reaching to $\boldsymbol{m_f}^l$, $l = 1$ to 7

Therefore we can see that the states converge to the final ones monotonically.

However, in the distributed method the markings of buffers are not controlled as in the centralized method, thus have more "freedom". Therefore we may not be able to conclude which one is better in general, just considering the time spent on the trajectories. Fig. 8.14 shows the state evolutions of buffer places (we take p_1, p_{16} and p_{45} as examples). We can observe that the markings of buffer places may reach different values: by using the centralized MPC the final markings specified in $\boldsymbol{m}_f$ are reached; but by using the distributed MPC, the markings of buffer places may reach some different (non-negative) values since here we only consider the convergence to the final states of subsystems. By using the distributed MPC the marking trajectory of buffer places, e.g., p_{45} in Fig. 8.14(b), is no longer monotone. For instance, after 31 time steps, the marking of p_{45} oscillates around the value of 0.1. This is because subsystem S^6 reaches its final state at 30 time steps (see Fig. 8.13(d)), then optimization problem (6.11) should be solved in S^6 and it tries to put more tokens to its (unique) output buffer place p_{45} when its marking $\boldsymbol{m}_k[p_{45}] \leq \epsilon_1 = 0.1$ (see cost function (6.9) and Remark 6.3.1); at the same time, subsystem S^7 needs to consume tokens from its input buffer place p_{45} for evolving towards its final state.

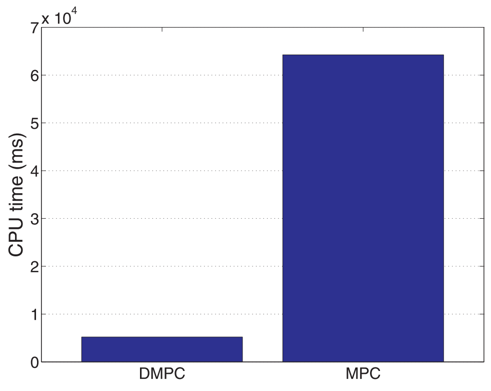


(a) The state evolution of p_1 and p_{16}

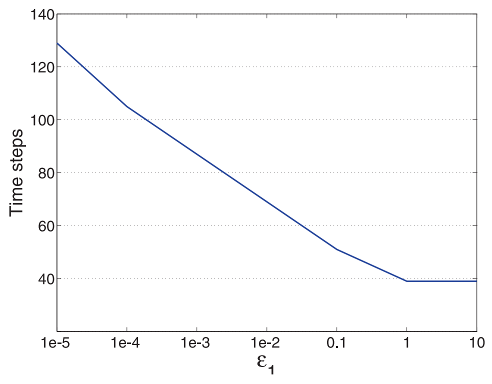


(b) The state evolution of p_{45}

Figure 8.14: The state evolution of some buffer places



(a)



(b)

Figure 8.15: (a) The CPU time of using centralized and distributed MPC. (b) The time steps of using the distributed MPC with different ϵ_1

By using the distributed MPC controller given in Algorithm 9, each subsystem S^l tries to put more tokens to its output buffer places $p_i \in B^{(l,\cdot)}$ (by solving the optimization problem (6.11)) only if it is not able to evolve towards its final state (or its final state has already been reached), and $\boldsymbol{m}_k[p_i] \leq \epsilon_1$ where ϵ_1 is assumed to be very small positive value to approximate $\boldsymbol{m}_k[p_i] = 0$ (in cost function (6.9)). One may think that when problem (6.11) is solved, if we could use a larger value for ϵ_1, i.e., if we could also try to put tokens into those output buffer places of S^l that are not "nearly emptied", to make its neighboring subsystems evolve faster. However, in this case the convergence to final states may not be guaranteed when two neighboring subystems are both solving problem (6.11) and there exist multiple input/output buffers between them. In this particular example, subsystems can converge to their final states faster if a larger value for ϵ_1 is used, shown in Fig. 8.15(b). Notice that when $\epsilon_1 \geq 1$, its number of time steps required to reach final states (39 time steps) is almost the same as using the centralized MPC (38 time steps).

8.7 Conclusions

In this chapter, we carry out several case studies to illustrate the control methods proposed in this thesis for the target marking control problem of TCPNs under infinite server semantics.

In the first three case studies, we focus on the centralized ON/OFF based control methods, aiming at driving the system to the final state and minimizing the time spent on state evolution. It is shown that an advantage of the ON/OFF based controllers is the low computational complexity: in all the case studies (simulations results shown in Table A.1–A.8), the ON/OFF+ and B-ON/OFF controllers have lower computational costs than the approaching minimum-time controller proposed in [5]; while for the MPC-ON/OFF controller with a small time horizon N, its computational cost may be also not higher (as for the systems in case studies 1–3). At the same time, reasonable numbers of time steps for reaching the final state can be obtained. The standard ON/OFF controller is the most suitable choice for a CF net system, ensuring low computational complexity and minimum-time. For non-CF nets, some characteristics of the system may be helpful in choosing an appropriate method. In particular, if the flows (depending on markings) of conflicting transitions are very different, the B-ON/OFF controller may obtain a smaller number of time steps than the ON/OFF+ controller. The approaching minimum-time controller [5] may not be a good choice if there are some places with very small markings (because in this approach, between each pair of adjacent states of the trajectory the firing speed is constant and determined by the one with a smaller flow); in those cases the ON/OFF based methods usually can achieve better results. It may be interesting to point out that in "slow" practical systems like logistics or manufacturing systems, the operational time may be much larger than the computational time for the control laws (considering the very low computational complexity of ON/OFF based methods; including the MPC-ON/OFF controller with small time horizon); therefore, at each operation instant we could compute the control laws by using several of the methods, and then choose the best one to apply.

In the last case study, we apply the distributed MPC controller proposed in Chapter 6. The other decentralized control methods proposed in Chapter 5 and contribution [4] are not applicable to this example, because the net or some subnets are not CF or mono-T-semiflow. We show that by using the distributed MPC controller given by Algorithm 9, all the subsystems reach their final state in finite time (but not minimum-time in general), while all the buffer places shared by subsystems are always in legal non-negative states.

Chapter 9

Conclusions and Future works

Many man-made systems, such as manufacturing, logistics, telecommunication or traffic systems, can be "naturally" viewed as Discrete Event Dynamic Systems (DEDS). Nevertheless, large populations or heavy traffic frequently appear and they may suffer from the classical *state explosion* problem. In order to overcome this problem, *fluidization* can be applied, obtaining the fluid relaxation of the original discrete model. Continuous Petri nets (CPNs) are a fluid approximation of discrete Petri nets (PNs), a well known formalism for DEDS. It is obtained by removing the integrality constraints on the firings of transitions, thus on the markings (states). One key benefit of using CPNs is that, most frequently, it leads to a substantial reduction in the computational costs. For instance, several techniques based on integer programming in the discrete model may be solved using linear programming in the fluid case. Among other interesting issues, many works can be found in the literature considering the modelling, analysis, observability and control of CPNs.

In this thesis we have focused on the control of timed continuous Petri nets (TCPNs), in which time interpretations are associated to transitions. Depending on how the flow of transitions is defined, there exist different server semantics and we assume the *infinite server semantics* (variable speed) because it has been observed to usually obtain better approximations of discrete PNs in general, and always under some particular conditions (net subclasses).

Regarding the control problem, the first question is what to control? We assume that control actions are applied to *slow down* the firing of transitions [89], i.e., to reduce the flow. We have considered two interesting control problems in this thesis:

- The first problem is called *target marking control*, where the objective is to drive the system (as fast as possible) from an initial state $\boldsymbol{m}_0$ to a desired final state $\boldsymbol{m}_f$. It is similar to the *set-point* control problem in a general continuous-state system. By considering the CPNs as a relaxation of discrete models, the continuous state can be viewed as an approximation of the average state in the original discrete system.

- The second problem is called *optimal flow control*, in which the objective is to drive the system to an optimal flow (obtained in a "convex final region"), without *a priori* knowledge of a specific final state. Usually, these final states with the optimal flow are not unique and they belong to a convex region. In this work, we are interested in reaching as fast as possible the maximal flow, what is frequently desirable in practical systems.

In both control problems, we address the minimum-time evolution, a problem that is close to the minimization of the *makespan* in manufacturing (time difference between the start and finish of a sequence of jobs or tasks). In many existing control methods, the computational costs may grow very quickly (for example, the approaching minimum-time controller [5], because computationally expensive BPPs should be solved), even exponentially (for example, the affine control [102], because the number of vertices increases exponentially on the number of places), with respect to the size of the net system. In this work we use an ON/OFF strategy, which is shown to be very efficient. We consider both centralized and decentralized settings for the target marking control problem.

Regarding the *centralized* methods we have developed several (heuristic) minimum-time controllers based on the ON/OFF strategy, which is frequently used in minimum-time problems:

- **ON/OFF controller**: This controller is specially suitable for Choice-Free (CF) nets. A interesting property of CF nets is that they are structurally persistent, i.e., for any initial state ($\boldsymbol{m}_0$), the enabling degree of a transition will not be decreased by firing other ones; furthermore, the reaching of one (final) state does not depend on the firing orders (speeds) of transitions, and it is only relevant to the required firing count vector. We have shown that the desired final state can be reached in minimum-time by using a simple ON/OFF strategy: fire every transition as fast as possible (ON) until an upper bound, given by the *minimal firing count vector*, is reached; then simply block it (OFF).

- **ON/OFF+ controller**: In the case that the net is not CF, we may not be able to apply the ON/OFF controller, because its "greedy" strategy of firing *conflicting* transitions may bring some "blocked" situations, even for live and bounded net systems. The ON/OFF+ controller overcomes this problem by forcing proportional firings of conflicting transitions according to a given firing count vector; while for the persistent transitions, the standard ON/OFF strategy is applied.

- **B-ON/OFF controller**: The ON/OFF+ controller requires very low computation costs, but it may have a drawback in some cases where the flows of conflicting transitions are very different: if conflicting transitions fire proportionally, then obviously, the overall time spent for firing a given firing count vector to reach $\boldsymbol{m}_f$ is determined by the "slower" ones. In the B-ON/OFF controller, this problem is handled by adding a "balancing process", i.e., we first fire the "faster" transitions and block the "slower" transitions for a period of time, until they get *balanced* (if they can). After that, we simply apply the

ON/OFF+ controller. In this way, the time spent for reaching the final state may be decreased, comparing with that of the ON/OFF+ controller.

- **MPC-ON/OFF controller**: Model Predictive Control (MPC) has been widely applied in industry for process control. Usually it is used for optimizing trajectory by solving certain optimization problems at each time step. In this work, we have tried to combine the MPC scheme with the ON/OFF strategy, obtaining a heuristics for approaching minimum-time target marking control. In particular, conflicts are solved by using the MPC approach; while the ON/OFF strategy is applied to persistent transitions.

All the proposed ON/OFF-based controllers ensure the *convergence*. For CF nets, the standard ON/OFF controller gives a minimum-time evolution; while other controllers are heuristics for minimum-time target marking control of general net systems and, we have seen in examples that reasonably small numbers of time steps for reaching the final state can be obtained. A main advantage of using the ON/OFF strategy is its low computational complexity. In ON/OFF, ON/OFF+ and B-ON/OFF controllers, only simple linear programming problems need to be solved at each time step. The MPC-ON/OFF controller may be computationally more expensive (a QPP problem is involved at each time step) and the computational cost grows fast with respect to the time horizon N. For example, for the net system in Fig. 4.11 (simulation results shown in Table A.1–A.4) the required CPU time increases about 50 times when N increases from 1 to 10. Nevertheless, even if a smaller number of time steps may be obtained by using a larger N, from our simulations, in some cases the improvement is not very significant; and even a worse result may be obtained using a larger N (e.g., in the case of setting $\mathbf{s}.1$) for the net system in Fig. 4.11); so using a small N may be a reasonable choice.

Decentralized control methods become interesting for large scale and usually physically distributed systems, for which centralized control may be difficult to implement. The existing work related to the decentralized control of TCPNs is still very limited, in this thesis we have proposed two approaches:

- **Decentralized minimum-time control of CF nets**: We assume that large scale systems are cut into subsystems through sets of buffer places (intersections). The idea is that we first compute the local control laws (minimal firing count vectors) separately in all subsystems, then we reach an agreement among them to get globally admissible control laws. In order to ensure that identical behaviors (firing sequences) of the original system are obtained in subsystems, we have employed several reduction rules to build abstractions of the missing parts of disconnected subsystems; for reaching the agreement among local control laws, a high level coordinator is introduced. When globally admissible control laws are obtained, the ON/OFF controller can be implemented independently in any subsystem, obtaining a minimum-time evolution.
- **Distributed MPC control of general nets**: For general net systems, the decentralized control method proposed for CF nets may not be applicable and

one reason is that we may not be able to achieve an agreement among local control laws. We have proposed a distributed control method for general nets based on the MPC approach. First, we present a *centralized* MPC controller, in which the state evolution is constrained to be inside an interior convex subset of the reachability space and thus, the convergence to the final state can be guaranteed. This approach is less constrained than the methods proposed in [5] and [64], in which linear trajectories were considered. Later, we have proposed a distributed MPC control algorithm. Similarly to the previous method for CF nets, we assume that subsystems are connected by sets of *buffer* places. A key issue in the distributed setting concerns the strict positiveness of the markings of buffer places: an alternative optimization problem is considered to recover from the situations where the markings of some buffer places are converging to zero.

In the decentralized control method for CF nets, we construct the *complemented subsystems* by using several reduction rules and the identical behaviros of the original system are preserved. Let us point out that those obtained complemented subsystems may also be useful in other contexts, for example, throughput approximations in a distributed way (similar works have been done for discrete models, for example in [20] for marked graphs, and later in [75] for weighted T-systems). The distributed MPC control method proposed here is not designed for the minimum-time control, but it can be applied for general nets and ensures that all the subsystems converge to their final states. As a by-product, the MPC controller applied in each subsystem can also be used independently in centralized setting.

The (minimum-time) optimal flow control problem has been considered here for CF nets. Usually, we may obtain the optimal flow (in our case, the maximal flow) of the system in a set of steady states, belonging to a convex subset in the reachability space. Since we cannot uniquely determine a desired final state, even for MG (a subclass of CF nets) the minimum-time flow control becomes difficult. In this work, we have proposed a heuristic algorithm for CF nets, in which we compute the "best" firing count vector bringing the system to the maximal flow, according to an estimation of the number of time steps based on the current state and flow; and an ON/OFF firing strategy is applied. We also show that because of the persistency of CF nets, we can further reduce the time spent to reach the maximal flow by means of some additional firings.

Despite the many results that can be found in the literature about the control of TCPNs, and some new methods proposed in this thesis, there still remain many widely open issues that deserve more investigations. Among many others:

1) In the proposed methods, the ON/OFF strategy is applied to achieve minimum-time evolution. It could be also important to consider the minimum-time control that will satisfy additional conditions. For example, control laws that require minimal control energy could be interesting.

2) More extensive comparisons of the available control methods are necessary, providing more concrete criteria of selecting suitable methods in different situations. For this aim, certain benchmarks with systems of different net structures, markings and firing rates are desirable.

3) Furthermore, comparisons should be performed also using the discrete models. A control method that works well for the fluid model may not be a good choice when the control laws are interpreted and applied to control the underlying discrete one.

4) Most of the existing results, including this work, assume that all the transitions are controllable. A clear extension is to further consider partially controllable systems.

Bibliography

[1] H. Alla and R. David. Continuous and Hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8:159–188, 1998.

[2] E. Altman, T. Jiménez, and G. Koole. On the comparison of queueing systems with their fluid limits. *Probability in the Engineering and Informational Sciences*, 15(2):165–178, 2001.

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

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

[5] H. Apaydin-Ozkan, J. Júlvez, C. Mahulea, and M. Silva. Approaching Minimum Time Control of Timed Continuous Petri nets. *Nonlinear Analysis: Hybrid Systems*, 5(2):136–148, May 2011.

[6] M. Athans and P. Falb. *Optimal control: an introduction to the theory and its applications*. McGraw-Hill New York, 1966.

[7] A. Aydin. Decentralized structural control approach for Petri nets. *Control and Cybernetics*, 36(1):143–159, 2007.

[8] G. Balbo and M. Silva. *Performance Models for discrete Event Systems with Synchronizations Formalisms and Analysis Techniques*. KRONOS, Zaragoza, 1998.

[9] F. Balduzzi, G. Menga, and A. Giua. First-order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.

[10] F. Basile, A. Giua, and C. Seatzu. Supervisory Control of Petri Nets with Decentralized Monitor Places. In *2007 American Control Conference*, pages 4957 – 4962, New York, USA, 2007.

[11] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *Automatic Control, IEEE Transactions on*, 45(10):1864–1876, 2000.

[12] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[13] G. Berthelot and Lri-Iie. Checking properties of nets using transformations. *Advances in Petri Nets 1985: Lecture Notes in Computer Science*, 222:19–40, 1986.

[14] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[15] J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[16] M. Bramson. Stability of queueing networks. *Probability Surveys*, 5(169-345):1, 2008.

[17] E. Camacho and C. Bordons. *Model Predictive Control.* New York: Springer-Verlag, 2004.

[18] E. Camponogara and L. B. de Oliveira. Distributed optimization for model predictive control of linear-dynamic networks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(6):1331–1338, 2009.

[19] J. Campos, G. Chiola, and M. Silva. Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector. *IEEE Transactions on Software Engineering*, 17:117–125, 2 1991.

[20] J. Campos, J. Colom, H. Jungnitz, and M. Silva. Approximate Throughput Computation of Stochastic Marked Graphs. *IEEE Transactions on Software Engineering*, 20(7):526–535, 1994.

[21] C. Chen. *Linear system theory and design.* Oxford University Press, New York, 1984.

[22] X. Cheng and B. Krogh. Stability-constrained model predictive control. *Automatic Control, IEEE Transactions on*, 46(11):1816–1820, 2001.

[23] G. Chiola, C. Anglano, J. Campos, J. M. Colom, and M. Silva. Operational Analysis of Timed Petri Nets and Application to the Computation of Performance Bounds. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 161–174. Springer, 1995.

[24] P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

[25] R. David and H. Alla. *Discrete, Continuous and Hybrid Petri Nets.* Springer, Berlin, 2004. (Revised 2^{nd} edition, 2010).

[26] M. Dotoli, M. Fanti, A. Giua, and C. Seatzu. First-order hybrid Petri nets. An application to distributed manufacturing systems. *Nonlinear Analysis: Hybrid Systems*, 2(2):408–430, 2008.

[27] M. Dub, G. Pipan, and Z. Hanzálek. Stock optimization of a kanban-based assembly line. In *Proc. of the 12th Int. Conf. on Flexible Automation and Intelligent Manufacturing*, 2002.

[28] J. Ezpeleta, J. M. Couvreur, and M. Silva. A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Coloured Petri Nets. pages 126–147.

[29] E. Fraca, J. Júlvez, and M. Silva. Marking homothetic monotonicity and fluidization of untimed Petri nets. In *11th Int. Workshop on Discrete Event Systems, WODES 2012*, Guadalajara, Mexico, October 2012.

[30] L. Ghomri and H. Alla. Continuous flow systems and control methodology using Hybrid Petri nets. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 419–424. IEEE, 2011.

[31] A. Giua, F. DiCesare, and M. Silva. Petri Net supervisors for generalized mutual exclusion constraints. In *12th IFAC World Congress*, pages 267–270, Sidney, Australia, 1993.

[32] L. Habets, P. Collins, and J. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, 2006.

[33] L. Habets and J. van Schuppen. A controllability result for piecewise-linear hybrid systems. In *Proceedings of European Control Conference (ECC2001)*, pages 3870–3873, 2001.

[34] L. Habets and J. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40(1):21–35, 2004.

[35] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. *Formal Methods for Computational Systems Biology*, pages 215–264, 2008.

[36] S. Hennequin, D. Lefebvre, and A. El-Moudni. Fuzzy control of variable speed continuous Petri nets. In *38th IEEE Conference on Decision and Control*, volume 2, pages 1352–1356, Phoenix, USA, Dec. 1999.

[37] K. Hiraishi. Performance evaluation of workflows using continuous Petri nets with interval firing speeds. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, 91(11):3219, 2008.

[38] L. E. Holloway, B. H. Krogh, and A. Giua. A Survey of Petri Net Methods for Controlled Discrete Event Systems. *Discrete Event Dynamic Systems*, 7(2):151–190, 1997.

[39] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1):184–201, 1998.

[40] M. V. Iordache and P. J. Antsaklis. Decentralized Supervision of Petri Nets. *IEEE Transactions on Automatic Control*, 51(2):376–391, 2006.

[41] M. V. Iordache and P. J. Antsaklis. *Supervisory control of concurrent systems: A Petri net structural approach.* Birkhauser, 2006.

[42] L. Jiao and T. Cheung. Characterizing liveness monotonicity for weighted Petri nets in terms of siphon-based properties. *International Journal of Foundations of Computer Science*, 14(4):641–658, 2003.

[43] E. Jiménez, J. Júlvez, L. Recalde, and M. Silva. On controllability of timed continuous Petri net systems: the join free case. In *Proc. of the 44th IEEE Conf. on Decision and Control (Joint CDC-ECC)*, Seville, 2005.

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

[45] X. Jing, L. Recalde, and M. Silva. Tracking control of timed continuous Petri net systems under infinite servers semantics. In *IFAC World Congress*, 2008.

[46] J. Júlvez. *Algebraic Techniques for the Analysis and Control of Continuous Petri Nets.* PhD thesis, University of Zaragoza, 2004.

[47] J. Júlvez, E. Jiménez, L. Recalde, and M. Silva. On observability and design of observers in timed continuous Petri net systems. *Automation Science and Engineering, IEEE Trans. on*, 5(3):532–537, July 2008.

[48] J. Júlvez, C. Mahulea, and C. Vázquez. SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems*, 6(2):806–817, 2012.

[49] J. Júlvez, L. Recalde, and M. Silva. On reachability in autonomous continuous petri net systems. In *Applications and Theory of Petri Nets 2003*, pages 221–240. Springer, 2003.

[50] J. Júlvez, L. Recalde, and M. Silva. Steady–state performance evaluation of continuous mono-T-semiflow Petri nets. *Automatica*, 41(4):605–616, 2005.

[51] R. Kara, M. Ahmane, J. Loiseau, and S. Djennoune. Constrained regulation of continuous Petri nets. *Nonlinear Analysis: Hybrid Systems*, 3(4):738–748, 2009.

[52] S. Keerthi and E. Gilbert. Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints. *IEEE Transactions on Automatic Control*, 32(5):432–435, 1987.

[53] C. Kelling. Timenet-sim-a parallel simulator for stochastic petri nets. In *Simulation Symposium, 1995., Proceedings of the 28th Annual*, pages 250–258. IEEE, 1995.

[54] A. Lazar. Optimal flow control of a class of queueing networks in equilibrium. *IEEE Transactions on Automatic Control*, 28(11):1001 – 1007, 1983.

[55] D. Lefebvre. Optimal flow control for manufacturing systems modelled by continuous Petri nets. In *39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[56] D. Lefebvre. About the stochastic and continuous Petri nets equivalence in the long run. *Nonlinear Analysis: Hybrid Systems*, 5(3):394–406, 2011.

[57] D. Lefebvre. Approximation of the asymptotic mean marking of SPNs with contPNs. *Nonlinear Analysis: Hybrid Systems*, 6(4):972–987, 2012.

[58] D. Lefebvre, C. Delherm, E. Leclercq, and F. Druaux. Some contributions with Petri nets for the modelling, analysis and control of HDS. *Nonlinear Analysis: Hybrid Systems*, 1:451–465, 2007.

[59] D. Lefebvre and E. Leclercq. Piecewise constant timed continuous PNs for the steady state estimation of stochastic PNs. *Discrete Event Dynamic Systems*, 22(2):179–196, 2012.

[60] Z. Li, M. Zhou, and N. Wu. A Survey and Comparison of Petri Net-Based Deadlock Prevention Policies for Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man and Cybernetics - PART C: APPLICATIONS AND REVIEWS*, 38(2):173–188, 2008.

[61] Z. W. Li and M. C. Zhou. Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(1):38–51, 2004.

[62] G. F. List and M. Cetin. Modeling traffic signal control using Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):177–187, 2004.

[63] C. Mahulea. *Timed Continuous Petri Nets: Quantitative Analysis, Observability and Control.* PhD thesis, University of Zaragoza, 2007.

[64] C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. Optimal model predictive control of Timed Continuous Petri nets. *IEEE Trans. on Automatic Control*, 53(7):1731 – 1735, August 2008.

[65] C. Mahulea, A. Ramírez, L. Recalde, and M.Silva. Steady state control reference and token conservation laws in continuous Petri net systems. *IEEE Transactions on Automation Science and Engineering*, 5(2):307–320, April 2008.

[66] C. Mahulea, L. Recalde, and M. Silva. Basic Server Semantics and Performance Monotonicity of Continuous Petri Nets. *Discrete Event Dynamic Systems: Theory and Applications*, 19(2):189 – 212, June 2009.

[67] C. Mahulea, L. Recalde, and M. Silva. Observability of continuous Petri nets with infinite server semantics. *Nonlinear Analysis: Hybrid Systems*, 4(2):219–232, 2010.

[68] C. Mahulea, C. Seatzu, M. Cabasino, and M. Silva. Fault Diagnosis of Discrete-Event Systems using Continuous Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 42(4):970 – 984, 2012.

[69] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with generalized stochastic Petri nets.* 1995, John Wiley and Sons.

[70] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[71] J. Merseguer and J. Campos. Software performance modeling using UML and Petri nets. *Performance Tools and Applications to Networked Systems*, pages 265–289, 2004.

[72] D. Mitra. Stochastic theory of a fluid model of producers and consumers coupled by a buffer. *Advances in Applied Probability*, pages 646–676, 1988.

[73] M. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9):913–917, 1982.

[74] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[75] C. Pérez-Jiménez, J. Campos, and M. Silva. Approximate Throughput Computation of Stochastic Weighted T-Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 37(3):431 – 444, 2007.

[76] C. A. Petri. *Communication With Automata.* Supplement 1 to Technical Report RADC-TR-65-377, Vol. 1, Griffiss Air Force Base, New York 1966.

[Originally published in German: Kommunikation mit Automaten, University of Bonn], 1962.

[77] L. S. Pontryagin. *Mathematical theory of optimal processes*, volume 4. CRC, 1987.

[78] S. Rajagopal, V. G. Kulkami, and J. Shaler Stidham. Optimal flow control of a stochastic fluid-flow system. *IEEE Journal on Selected Areas in Communications*, 13(7):1219 – 1228, 1995.

[79] L. Recalde, S. Haddad, and M. Silva. Continuous Petri Nets: Expressive Power and Decidability Issues. *Int. Journal of Foundations of Computer Science*, 21(2):235–256, 2010.

[80] L. Recalde and M. Silva. Petri nets fluidification revisited semantics and steady state. *APII JESA*, 35(4):435–449, 2001.

[81] L. Recalde, E. Teruel, and M. Silva. Modeling and Analysis of Sequential Processes that Cooperate Through Buffers. *IEEE Transactions on Robotics and Automation*, 14(2):267–276, 1998.

[82] L. Recalde, E. Teruel, and M. Silva. On Linear Algebraic Techniques for Liveness Analysis of P/T Systems. *Journal of Circuits Systems and Computers*, 8(1):223–265, 1998.

[83] L. Recalde, E. Teruel, and M. Silva. Autonomous Continuous P/T systems. In J. K. S. Donatelli, editor, *Application and Theory of Petri Nets 1999*, volume 1639 of *Lecture Notes in Computer Science*, pages 107–126. Springer, 1999.

[84] R. Roberto, R. Antonio, M. J. Alejandro, and R. Javier. Control Of Metabolic Systems Modeled with Timed Continuous Petri Nets. In *Proceedings of the Workshops of the 31st International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2010) and of the 10th International Conference on Application of Concurrency to System Design (ACSD 2010)*, pages 87–102, Braga, Portugal, June 2010.

[85] R. Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, 19(5):723–731, 2009.

[86] K. Shin and N. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[87] M. Silva, J. Júlvez, C. Mahulea, and C. R. Vázquez. On fluidization of discrete event models: observation and control of continuous Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 21(4):427–497, 2011.

[88] M. Silva and L. Recalde. Petri nets and integrality relaxations: A view of continuous Petri net models. *IEEE Trans. on Systems, Man, and Cybernetics*, 32(4):314–327, 2002.

[89] M. Silva and L. Recalde. On fluidification of Petri net models: from discrete to hybrid and continuous models. *Annual Reviews in Control*, 28(2):253–266, 2004.

[90] M. Silva, E. Teruel, and J. M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Net Systems. *Lecture Notes in Computer Science*, 1491:309–373, 1998.

[91] B. Stewart, A. Venkat, J. Rawlings, S. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59(8):460–469, 2010.

[92] E. Teruel, P. Chrzastowski-Wachtel, J. Colom, and M. Silva. On weighted T-systems. In K. Jensen, editor, *Application and Theory of Petri nets. LNCS*, volume 616, pages 348–367, Berlin, Germany, 1992.

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

[94] E. Teruel and M. Silva. Structure theory of equal conflict systems. *Theoretical Computer Science*, 153(1):271–300, 1996.

[95] K. Trivedi and V. Kulkarni. FSPNs: fluid stochastic Petri nets. *Application and Theory of Petri Nets 1993*, pages 24–31, 1993.

[96] C. Vázquez. *Fluidization, Controllability and Control of Timed Continuous Petri Nets*. PhD thesis, University of Zaragoza, 2011.

[97] C. Vázquez, A. Ramírez, L. Recalde, and M. Silva. On Controllability of Timed Continuous Petri Nets. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computational and Control (HSCC'08)*, volume 4981, pages 528—-541, Saint Louis, USA, July 2008. Springer Berlin / Heidelberg.

[98] C. Vázquez and M. Silva. Performance control of Markovian Petri nets via fluid models: A stock-level control example. In *IEEE International Conference on Automation Science and Engineering*, pages 30–36, 2009.

[99] C. Vázquez and M. Silva. Piecewise-linear constrained control for timed continuous Petri nets. In *Proc. of the 48th IEEE Conf. on Decision and Control (CDC)*, 2009.

[100] C. Vázquez and M. Silva. Timing and liveness in continuous Petri nets. *Automatica*, 47:283–290, 2011.

[101] C. Vázquez and M. Silva. Stochastic Continuous Petri Nets: An approximation of Markovian Net Models. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(3):641–653, 2012.

[102] C. Vázquez, J. van Schuppen, and M. Silva. A modular-coordinated control for continuous Petri nets. In *18th IFAC World Congress*, Milan, Italy, August 2011.

[103] L. Wang, C. Mahulea, J. Júlvez, , and M. Silva. Minimum-time Decentralized Control of Choice-Free Continuous Petri Nets. *Nonlinear Analysis: Hybrid Systems*, 7(1):39–53, 2013.

[104] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Minimum-time Control for Structurally Persistent Continuous Petri Nets. In *49th IEEE Conference on Decision and Control*, pages 2771–2776, Atlanta, Georgia USA, December 2010.

[105] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Decentralized Control of Large Scale Systems Modeled with Continuous Marked Graphs. In *18th IFAC World Congress*, Milan, Italy, Aug. 2011.

[106] L. Wang, C. Mahulea, J. Júlvez, and M. Silva. Control of continuous Petri nets using ON/OFF based method. In *WODES12: 11th Int. Workshop on Discrete Event Systems*, pages 47–52, Guadalajara, Mexico, October 2012.

[107] L. Wang, C. Mahulea, and M. Silva. Distributed Model Predictive Control of Timed Continuous Petri nets. In *52nd IEEE Conference on Decision and Control (CDC2013)*, Italy, December 2013. submitted.

[108] L. Wang, C. Mahulea, and M. Silva. Minimum-Time Flow Control of Timed Continuous Choice-Free Nets. In *ECC13: European Control Conference*, Zurich, Switzerland, July 2013. to appear.

[109] W. Wang, M. Palaniswami, and S. Low. Optimal flow control and routing in multi-path networks. *Performance Evaluation*, 52(2):119–132, 2003.

[110] G. Xie and L. Wang. Controllability and stabilizability of switched linear-systems. *Systems & Control Letters*, 48(2):135–155, 2003.

[111] A. Zimmermann, D. Rodríguez, and M. Silva. A two phase optimization method for Petri net models of manufacturing systems. *Journal of Intelligent Manufacturing*, 12(5):409–420, 2001.

Appendix A

Simulation Results

Table A.1: Simulation results of the net system in Fig. 4.11. Setting **s**.1): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	101	847	
ON/OFF+	94	41	
B-ON/OFF	91	130	$d = 1$
B-ON/OFF	91	136	$d = 2$
B-ON/OFF	94	141	$d = 5$
B-ON/OFF	94	139	$d = 10$
B-ON/OFF	94	138	$d = 15$
B-ON/OFF	94	142	$d = 20$
MPC-ON/OFF	91	1,159	$N = 1$, r=1000, q=1000
MPC-ON/OFF	91	877	$N = 1$, r=1000, q=100
MPC-ON/OFF	91	955	$N = 1$, r=1000, q=1
MPC-ON/OFF	95	5,546	$N = 3$, r=1000, q=1000
MPC-ON/OFF	95	5,619	$N = 3$, r=1000, q=100
MPC-ON/OFF	97	7,678	$N = 3$, r=1000, q=1
MPC-ON/OFF	94	11,188	$N = 5$, r=1000, q=1000
MPC-ON/OFF	95	11,369	$N = 5$, r=1000, q=100
MPC-ON/OFF	91	11,811	$N = 5$, r=1000, q=1
MPC-ON/OFF	93	54,311	$N = 10$, r=1000, q=1000
MPC-ON/OFF	94	53,818	$N = 10$, r=1000, q=10
MPC-ON/OFF	94	50,784	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF ($d = 1$ or 2) or the MPC-ON/OFF controller (with $N = 1$ or 5). For the B-ON/OFF controller, smaller numbers of time steps are obtained using smaller values of d (a small value of d implies that the “slower” transitions in a conflict will keep blocked until their flows get very “balanced” with the “faster” ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the “slower” transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.2: Simulation results of the net system in Fig. 4.11. Setting **s**.2): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.001\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.301\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	176	465	
ON/OFF+	954	410	
B-ON/OFF	132	197	$d = 1$
B-ON/OFF	132	192	$d = 2$
B-ON/OFF	196	287	$d = 5$
B-ON/OFF	258	393	$d = 10$
B-ON/OFF	290	437	$d = 15$
B-ON/OFF	335	504	$d = 20$
MPC-ON/OFF	165	1,942	$N = 1$, r=1000, q=1000
MPC-ON/OFF	158	1,687	$N = 1$, r=1000, q=100
MPC-ON/OFF	149	2,697	$N = 1$, r=1000, q=1
MPC-ON/OFF	165	8,005	$N = 3$, r=1000, q=1000
MPC-ON/OFF	164	8,048	$N = 3$, r=1000, q=100
MPC-ON/OFF	163	7,448	$N = 3$, r=1000, q=1
MPC-ON/OFF	162	14,638	$N = 5$, r=1000, q=1000
MPC-ON/OFF	162	14,593	$N = 5$, r=1000, q=100
MPC-ON/OFF	145	16,240	$N = 5$, r=1000, q=1
MPC-ON/OFF	159	80,193	$N = 10$, r=1000, q=1000
MPC-ON/OFF	159	80,417	$N = 10$, r=1000, q=10
MPC-ON/OFF	159	86,426	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF controller ($d = 1$ or 2). For the B-ON/OFF controller, smaller numbers of time steps are obtained using smaller values of d (a small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly. For the MPC-ON/OFF controller, smaller numbers of time steps are obtained by using larger weights on matrix $\boldsymbol{R}$ and smaller weights on matrix $\boldsymbol{Q}$; the numbers of time steps are slightly reduced by using larger N.

Table A.3: Simulation results of the net system in Fig. 4.11. Setting **s**.3): $\Theta = 0.1$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 3\ 2.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	94	352	
ON/OFF+	76	34	
B-ON/OFF	78	121	$d = 1$
B-ON/OFF	78	120	$d = 2$
B-ON/OFF	78	121	$d = 5$
B-ON/OFF	76	114	$d = 10$
B-ON/OFF	76	114	$d = 15$
B-ON/OFF	76	115	$d = 20$
MPC-ON/OFF	94	958	$N = 1$, r=1000, q=1000
MPC-ON/OFF	94	1,010	$N = 1$, r=1000, q=100
MPC-ON/OFF	94	1,067	$N = 1$, r=1000, q=1
MPC-ON/OFF	93	9,800	$N = 3$, r=1000, q=1000
MPC-ON/OFF	92	5,178	$N = 3$, r=1000, q=100
MPC-ON/OFF	93	7,394	$N = 3$, r=1000, q=1
MPC-ON/OFF	90	13,958	$N = 5$, r=1000, q=1000
MPC-ON/OFF	84	14,726	$N = 5$, r=1000, q=100
MPC-ON/OFF	84	15,482	$N = 5$, r=1000, q=1
MPC-ON/OFF	75	52,803	$N = 10$, r=1000, q=1000
MPC-ON/OFF	78	57,987	$N = 10$, r=1000, q=10
MPC-ON/OFF	76	56,352	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the MPC-ON/OFF controller (with $N = 10$). The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.4: Simulation results of the net system in Fig. 4.11. Setting $\mathbf{s}.4$): $\Theta = 0.02$, $\boldsymbol{m}_0 = [1\ 2\ 1.4\ 1.5\ 1.1\ 1.1\ 1.1\ 5\ 1.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 1.7\ 1.2\ 1.2\ 1.5\ 1.3\ 3\ 3.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	122	2,546	
ON/OFF+	126	55	
B-ON/OFF	128	200	$d = 1$
B-ON/OFF	128	195	$d = 2$
B-ON/OFF	128	195	$d = 5$
B-ON/OFF	126	191	$d = 10$
B-ON/OFF	126	192	$d = 15$
B-ON/OFF	126	195	$d = 20$
MPC-ON/OFF	133	1,458	$N = 1$, r=1000, q=1000
MPC-ON/OFF	133	1,441	$N = 1$, r=1000, q=100
MPC-ON/OFF	115	1,283	$N = 1$, r=1000, q=1
MPC-ON/OFF	131	5,855	$N = 3$, r=1000, q=1000
MPC-ON/OFF	131	6,943	$N = 3$, r=1000, q=100
MPC-ON/OFF	131	6,386	$N = 3$, r=1000, q=1
MPC-ON/OFF	128	13,580	$N = 5$, r=1000, q=1000
MPC-ON/OFF	130	13,539	$N = 5$, r=1000, q=100
MPC-ON/OFF	130	16,016	$N = 5$, r=1000, q=1
MPC-ON/OFF	126	92,728	$N = 10$, r=1000, q=1000
MPC-ON/OFF	126	90,033	$N = 10$, r=1000, q=10
MPC-ON/OFF	125	90,781	$N = 10$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the approaching minimum-time controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.5: Simulation results of the net system in Fig. 8.5. $\boldsymbol{m}_0 = [0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 4\ 4\ 2\ 5\ 1\ 2\ 1\ 2]^T$, $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.8125\ 1.083\ 0.8125\ 0.55]^T$, $\boldsymbol{\lambda} = [10\ 10\ 2\ 2.5\ 2.5\ 10\ 10\ 1.25\ 2.5\ 2.5\ 2.5\ 0.5]^T$, $\boldsymbol{\sigma} = [2.345\ 2.368\ 2.08\ 1.87\ 1.66\ 2.578\ 2.6\ 2.08\ 1.87\ 1.66\ 1.45\ 0]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j, j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	311	88,505	
ON/OFF+	457	275	
B-ON/OFF	217	443	$d = 1$
B-ON/OFF	209	426	$d = 2$
B-ON/OFF	210	431	$d = 5$
B-ON/OFF	216	448	$d = 10$
B-ON/OFF	234	491	$d = 15$
B-ON/OFF	280	596	$d = 20$
MPC-ON/OFF	220	4,815	$N = 1$, r=1000, q=1000
MPC-ON/OFF	221	4,994	$N = 1$, r=1000, q=100
MPC-ON/OFF	221	5,023	$N = 1$, r=1000, q=1
MPC-ON/OFF	220	29,115	$N = 3$, r=1000, q=1000
MPC-ON/OFF	220	28,947	$N = 3$, r=1000, q=100
MPC-ON/OFF	220	30,017	$N = 3$, r=1000, q=1
MPC-ON/OFF	219	92,156	$N = 5$, r=1000, q=1000
MPC-ON/OFF	219	94,743	$N = 5$, r=1000, q=100
MPC-ON/OFF	219	102,622	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF controller (with $d = 2$). For the B-ON/OFF controller, smaller numbers of time steps are obtained with smaller values of d (a small value of d implies that the "slower" transitions in a conflict will keep blocked until their flows get very "balanced" with the "faster" ones; then the proportional firing strategy (used in the ON/OFF+ controller) is applied to fire the "slower" transitions, and this strategy is more suitable when conflicting transitions have very similar flows); if d is very large, the B-ON/OFF controller is not sensitive to the difference of flows among conflicting transitions, and it is similar to applying the ON/OFF+ controller directly), but an exception happened when $d = 1$. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to time horizon N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.6: Simulation results of the net system in Fig. 8.6. $\boldsymbol{m}_0 = [0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 4\ 4\ 2\ 5\ 1\ 2\ 1\ 2]^T$, $\boldsymbol{m}_f = [0.0775\ 0.3875\ 0.31\ 0.31\ 0.0775\ 0.62\ 0.31\ 0.31\ 0.31\ 0.31\ 1.55\ 2.13\ 2.13\ 1.315\ 0.0775\ 0.58\ 1.083\ 1.045\ 0.55]^T$, $\boldsymbol{\lambda} = [10\ 10\ 2\ 2.5\ 2.5\ 10\ 10\ 1.25\ 2.5\ 2.5\ 2.5\ 0.5]^T$, $\boldsymbol{\sigma} = [2.345\ 2.368\ 2.08\ 1.87\ 1.66\ 2.578\ 2.6\ 2.08\ 1.87\ 1.66\ 1.45\ 0]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j, j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	316	55,608	
ON/OFF+	223	108	
B-ON/OFF	224	427	$d = 1$
B-ON/OFF	223	420	$d = 2$
B-ON/OFF	223	421	$d = 5$
B-ON/OFF	223	421	$d = 10$
B-ON/OFF	223	421	$d = 15$
B-ON/OFF	223	419	$d = 20$
MPC-ON/OFF	229	4,967	$N = 1$, r=1000, q=1000
MPC-ON/OFF	229	4,846	$N = 1$, r=1000, q=100
MPC-ON/OFF	228	5,018	$N = 1$, r=1000, q=1
MPC-ON/OFF	230	31,281	$N = 3$, r=1000, q=1000
MPC-ON/OFF	229	30,536	$N = 3$, r=1000, q=100
MPC-ON/OFF	229	31,914	$N = 3$, r=1000, q=1
MPC-ON/OFF	228	96,231	$N = 5$, r=1000, q=1000
MPC-ON/OFF	228	97,993	$N = 5$, r=1000, q=100
MPC-ON/OFF	228	99,749	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the ON/OFF+ controller or the B-ON/OFF controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps are not sensitive to time horizon N, or the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.7: Simulation results of the net system in Fig. 8.8. $\boldsymbol{m}_0 = [10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 5\ 0.1\ 0.1\ 5\ 5\ 5\ 5\ 5\ 5\ 5\ 5\ 8\ 0.1\ 2\ 2\ 2\ 2\ 2]^T$, $\boldsymbol{m}_f = [5.78\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 1.024\ 0.05122\ 4.756\ 2.049\ 0.05122\ 0.5122\ 0.05122\ 5.78\ 1.024\ 0.05122\ 0.5122\ 0.05122\ 1.024\ 0.05122\ 4.124\ 2.049\ 0.2561\ 4.637\ 4.124\ 3.1\ 4.637\ 4.124\ 4.637\ 4.124\ 4.844\ 7.844\ 0.2561\ 0.7634\ 0.05122\ 0.1512\ 0.1512\ 0.05122]^T$, $\boldsymbol{\lambda} = [50\ 5\ 50\ 2.5\ 50\ 50\ 1.25\ 50\ 5\ 50\ 2.5\ 50\ 5;50\ 2.5\ 2.5\ 50\ 1.25\ 10\ 1.25\ ]^T$, $\boldsymbol{\sigma} = [4.22\ 3.807\ 3.856\ 2.932\ 2.98\ 5.244\ 3.295\ 3.344\ 2.932\ 4.22\ 3.295\ 3.344\ 2.932\ 2.98\ 2.056\ 2.056\ 2.105\ 0.1561\ 0\ 0.1561]^T$, $\Theta = 0.01$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j,j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	300	138,826	
ON/OFF+	249	213	
B-ON/OFF	131	470	$d = 1$
B-ON/OFF	131	471	$d = 2$
B-ON/OFF	131	472	$d = 5$
B-ON/OFF	131	475	$d = 10$
B-ON/OFF	150	550	$d = 15$
B-ON/OFF	249	892	$d = 20$
MPC-ON/OFF	146	10,117	$N = 1$, r=1000, q=1000
MPC-ON/OFF	150	10,377	$N = 1$, r=1000, q=100
MPC-ON/OFF	149	10,852	$N = 1$, r=1000, q=1
MPC-ON/OFF	134	86,935	$N = 3$, r=1000, q=1000
MPC-ON/OFF	133	84,778	$N = 3$, r=1000, q=100
MPC-ON/OFF	134	83,968	$N = 3$, r=1000, q=1
MPC-ON/OFF	132	284,205	$N = 5$, r=1000, q=1000
MPC-ON/OFF	131	284,859	$N = 5$, r=1000, q=100
MPC-ON/OFF	132	276,821	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the B-ON/OFF ($d \leq 10$) or the MPC-ON/OFF controller (with $N = 5$). For the B-ON/OFF controller, smaller numbers of time steps are obtained with smaller values of d, when $d \leq 10$ the best result is obtained. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Table A.8: Simulation results of the net system in Fig. 8.10. $\boldsymbol{m}_0 = [5\ 0.1\ 0.1\ 0.1\ 10\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 7\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 3\ 3\ 3\ 1\ 1\ 1\ 1]^T$, $\boldsymbol{m}_f = [3.4\ 1.3\ 0.4\ 0.2\ 7.6\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 0.4\ 3.1\ 0.8\ 0.4\ 0.4\ 0.4\ 2.4\ 2\ 0.8\ 0.4\ 0.7\ 0.4\ 0.4\ 0.4]^T$, $\lambda = [0.3333\ 0.5\ 0.5\ 0.5\ 0.5\ 1\ 0.5\ 0.5\ 0.5\ 0.5\ 0.25\ 0.5\ 0.5\ 1\ 0.5\ 0.5\ 0.5\ 0.5\ 0.5\ 0.25]^T$, $\boldsymbol{\sigma} = [2.4\ 1.2\ 0.9\ 0.6\ 0.3\ 0\ 0.9\ 0.6\ 0.3\ 0\ 1.6\ 0.4\ 0.1\ 0\ 3.9\ 1.6\ 1.3\ 1\ 0.7\ 0]^T$, $\Theta = 0.03$. For the MPC-ON/OFF controller, the weight matrix $\boldsymbol{Q} = q \cdot \boldsymbol{I}^{|P|}$, $\boldsymbol{R}[j, j] = r, \forall t_j \in T_p$.

Control methods	**Time steps**	**CPU time** (ms)	**Parameters**
appro. min-time	279	60,691	
ON/OFF+	301	271	
B-ON/OFF	301	1,151	$d = 1$
B-ON/OFF	301	1,144	$d = 2$
B-ON/OFF	301	1,151	$d = 5$
B-ON/OFF	301	1,147	$d = 10$
B-ON/OFF	301	1,149	$d = 15$
B-ON/OFF	301	1,157	$d = 20$
MPC-ON/OFF	320	17,873	$N = 1$, r=1000, q=1000
MPC-ON/OFF	320	19,783	$N = 1$, r=1000, q=100
MPC-ON/OFF	320	21,440	$N = 1$, r=1000, q=1
MPC-ON/OFF	317	126,208	$N = 3$, r=1000, q=1000
MPC-ON/OFF	315	129,551	$N = 3$, r=1000, q=100
MPC-ON/OFF	316	137,814	$N = 3$, r=1000, q=1
MPC-ON/OFF	310	431,671	$N = 5$, r=1000, q=1000
MPC-ON/OFF	310	425,928	$N = 5$, r=1000, q=100
MPC-ON/OFF	310	443,439	$N = 5$, r=1000, q=1

Remark. The smallest number of time steps to reach the final state is obtained by applying the approaching minimum-time controller. The B-ON/OFF controller does not improve the result of the ON/OFF+ controller (which is already close to the best), and the numbers of time steps are not sensitive to the values of d. For the MPC-ON/OFF controller, the numbers of time steps can be reduced by using larger N; the numbers of time steps are not sensitive to the weights on matrix $\boldsymbol{R}$ and $\boldsymbol{Q}$.

Appendix B

More discussions of the MPC-ON/OFF controller

From the simulation results shown in Appendix. A we can observe that although the number of time steps obtained by using the MPC-ON/OFF controller is not very far from the best in general, but it requires more CPU time for computing the control laws. On the other hand, it seems difficult to know how to choose an appropriate parameters (horizon time, weight matrix) for reducing the time. One basic reason is that in this MPC based approach, we consider a cost function of the trajectory, which is difficult to be used for obtaining the minimum-time evolution. However, this method may be interesting when we consider both the time of evolution and the cost of state trajectory.

According to the cost function in (4.14), we can define the cost of a state $\boldsymbol{m}_k$ on the trajectory as:

$$c_k = (\boldsymbol{m}_k - \boldsymbol{m}_f)' \cdot \boldsymbol{Q} \cdot (\boldsymbol{m}_k - \boldsymbol{m}_f) - \boldsymbol{w}_k' \cdot \boldsymbol{R} \cdot \boldsymbol{w}_k$$

Then, the average cost of the trajectory can be computed by:

$$cost = \left(\sum_{k=1}^{s} c_k \right) / s \tag{B.1}$$

where s is the number of time steps required to reach $\boldsymbol{m}_f$, i.e., the sequence of states on the trajectory is $\boldsymbol{m}_0, \boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_s (= \boldsymbol{m}_f)$. Let us consider the example that is shown in Fig. 4.11 and we assume that in the MPC-ON/OFF method, the weight matrices are $Q = R = 1000 \cdot \boldsymbol{I}$, the simulation results are shown in Table B.1 - B.4. It can be observed that, in all the cases, the MPC-ON/OFF controller has very low trajectory cost, meanwhile the obtained numbers of time steps are not bad.

Table B.1: Simulation results of the net system in Fig. 4.11. Setting s.1): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$.

Control methods	**Time steps**	**CPU time** (ms)	**Trajectory costs** c	**Parameters**
ON/OFF+	94	41	137	
B-ON/OFF	91	130	75	$d = 1$
B-ON/OFF	91	136	77	$d = 2$
B-ON/OFF	94	141	137	$d = 5$
B-ON/OFF	94	139	137	$d = 10$
B-ON/OFF	94	138	137	$d = 15$
B-ON/OFF	94	142	137	$d = 20$
MPC-ON/OFF	91	1,159	7	$N = 1$
MPC-ON/OFF	95	5,546	6	$N = 3$
MPC-ON/OFF	94	11,188	6	$N = 5$

Table B.2: Simulation results of the net system in Fig. 4.11. Setting s.2): $\Theta = 0.01$, $\boldsymbol{m}_0 = [1\ 2\ 0.001\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.301\ 0.2\ 0.2\ 0.5\ 0.3\ 4.7\ 0.4]^T$, $\boldsymbol{\sigma} = [0.4\ 0\ 0.2\ 0.5\ 0.3\ 0.1\ 0]^T$.

Control methods	**Time steps**	**CPU time** (ms)	**Trajectory cost**	**Parameters**
ON/OFF+	954	410	617	
B-ON/OFF	132	197	115	$d = 1$
B-ON/OFF	132	192	115	$d = 2$
B-ON/OFF	196	287	140	$d = 5$
B-ON/OFF	258	393	217	$d = 10$
B-ON/OFF	290	437	251	$d = 15$
B-ON/OFF	335	504	297	$d = 20$
MPC-ON/OFF	165	1,942	35	$N = 1$
MPC-ON/OFF	165	8,005	35	$N = 3$
MPC-ON/OFF	162	14,638	35	$N = 5$

Table B.3: Simulation results of the net system in Fig. 4.11. Setting **s**.3): $\Theta = 0.1$, $\boldsymbol{m}_0 = [1\ 2\ 0.4\ 0.5\ 0.1\ 0.1\ 0.1\ 5\ 0.1]^T$, $\boldsymbol{m}_f = [0.6\ 1.8\ 0.7\ 0.2\ 0.2\ 0.5\ 0.3\ 3\ 2.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$.

Control methods	**Time steps**	**CPU time** (ms)	**Trajectory cost**	**Parameters**
ON/OFF+	76	34	2,288	
B-ON/OFF	78	121	1,856	$d = 1$
B-ON/OFF	78	120	1,856	$d = 2$
B-ON/OFF	78	121	1,856	$d = 5$
B-ON/OFF	76	114	2,288	$d = 10$
B-ON/OFF	76	114	2,288	$d = 15$
B-ON/OFF	76	115	2,288	$d = 20$
MPC-ON/OFF	94	958	2,248	$N = 1$
MPC-ON/OFF	93	9,800	1,750	$N = 3$
MPC-ON/OFF	90	13,958	1,453	$N = 5$

Table B.4: Simulation results of the net system in Fig. 4.11. Setting **s**.4): $\Theta = 0.02$, $\boldsymbol{m}_0 = [1\ 2\ 1.4\ 1.5\ 1.1\ 1.1\ 1.1\ 5\ 1.1]^T$, $\boldsymbol{m}_f =[0.6\ 1.8\ 1.7\ 1.2\ 1.2\ 1.5\ 1.3\ 3\ 3.1]^T$, $\boldsymbol{\sigma} = [2.1\ 1.7\ 1.9\ 2.2\ 2\ 1.8\ 0]^T$.

Control methods	**Time steps**	**CPU time** (ms)	**Trajectory cost**	**Parameters**
ON/OFF+	126	55	1,397	
B-ON/OFF	128	200	1,051	$d = 1$
B-ON/OFF	128	195	1,051	$d = 2$
B-ON/OFF	128	195	1,051	$d = 5$
B-ON/OFF	126	191	1,397	$d = 10$
B-ON/OFF	126	192	1,397	$d = 15$
B-ON/OFF	126	195	1,397	$d = 20$
MPC-ON/OFF	133	1,458	189	$N = 1$
MPC-ON/OFF	131	5,855	142	$N = 3$
MPC-ON/OFF	128	13,580	98	$N = 5$