Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Jiaxiang Yan

Entitled
Modeling, Monitoring and Optimization of Discrete Event Systems Using Petri Nets

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Lingxi Li
Chair

Brian King

Yaobin Chen

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Lingxi Li

Approved by: Brian King 04/10/2013
Head of the Graduate Program Date

MODELING, MONITORING AND OPTIMIZATION OF DISCRETE EVENT SYSTEMS USING PETRI NETS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jiaxiang Yan

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2013

Purdue University

Indianapolis, Indiana

To my parents Shijun and Longying, and my grandparents, for their love and support.

ACKNOWLEDGMENTS

In the first and foremost place, I would like to express my most sincere appreciation to my adviser, Professor Lingxi Li, for his outstanding guidance and unconditional support throughout my graduate study. It is Professor Li who introduced to me the fantastic world of discrete event systems. Without his instruction and encouragement, this thesis would not have reached its current form.

I am also grateful to many faculty members at IUPUI for their help and contribution to my thesis work. My committee member, Professor Brian King, was always a nice resource for discussion and inspiration. His experience in algorithm design and error control coding helped me to improve my algorithm structure and efficiency and enhance my fault-tolerant controller. His expertise in LaTex also did me a great favor in refurbishing my thesis. Another committee member, Professor Yaobin Chen, usually gave me inspiration to combine my research with industrial needs based on his rich industrial experience and insights. I also would like to thank Professor Eliza Du, Stanley Chien, and Dongsoo Kim. The various project and paper experience with them benefited me a lot in my thesis research.

I would like to thank all my group members, Dr. Maria Paola Cabasino, Harpreet, Adam, Omar, Mingye, and Ali, for their very useful suggestions on my thesis research. Dr. Cabasino is an expert in Petri nets. Her experience in fault diagnosis provided me plenty of suggestions on transition firing sequence reconstruction and fault-tolerant controller. Omar and I both focus on Petri net research. His effort in applying Petri nets to traffic system modeling lent me much inspiration on my own work. Harpreet and Adam are familiar with practical system control and helped me build my research direction. The discussion with Mingye, who focuses on vehicular communication and adaptive cruise control, provided many helpful advices on my research in intelligent transportation systems. Ali gave me many useful suggestions on my slides design and presentation skills.

I also want to acknowledge my family members and friends for their support, especially my officemates Pingge, Kai, Shan, Feng, Renran and Jie. They made my study at SL 023 and SL 063 productive and enjoyable. Special thanks go to Sherrie and Valerie for their help with administrative issues (and in the preparation of morning coffee and donuts to keep us energetic).

Finally, I would like to acknowledge IUPUI and TOYOTA for their financial support at various stages of my graduate research.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Yan, Jiaxiang. M.S.E.C.E., Purdue University, May 2013. Modeling, Monitoring and Optimization of Discrete Event Systems Using Petri Nets. Major Professor: Lingxi Li.

In last decades, the research of discrete event systems (DESs) has attracts more and more attention because of the fast development of intelligent control strategies. Such control measures combine the conventional control strategies with discrete decision-making processes which simulate human decision-making processes. Due to the scale and complexity of common DESs, the dedicated models, monitoring methods and optimal control strategies for them are necessary. Among various DES models, Petri nets are famous for the advantage in dealing with asynchronous processes. They have been widely applied in intelligent transportation systems (ITS) and communication technology in recent years. With encoding of the Petri net state, we can also enable fault detection and identification capability in DESs and mitigate potential human errors. This thesis studies various problems in the context of DESs that can be modeled by Petri nets. In particular, we focus on systematic modeling, asynchronous monitoring and optimal control strategies design of Petri nets.

This thesis starts by looking at the systematic modeling of ITS. A microscopic model of signalized intersection and its two-layer timed Petri net representation is proposed in this thesis, where the first layer is the representation of the intersection and the second layer is the representation of the traffic light system. Deterministic and stochastic transitions are both involved in such Petri net representation. The detailed operation process of such Petri net representation is stated. The improvement of such Petri net representation is also provided with comparison to previous models.

Then we study the asynchronous monitoring of sensor networks. An event sequence reconstruction algorithm for a given sensor network based on asynchronous observations of its state changes is proposed in this thesis. We assume that the sensor network is modeled as a Petri net and the asynchronous observations are in the form of state (token) changes at different places in the Petri net. More specifically, the observed sequences of state changes are provided by local sensors and are *asynchronous*, i.e., they only contain partial information about the ordering of the state changes that occur. We propose an approach that is able to partition the given net into several subnets and reconstruct the event sequence for each subnet. Then we develop an algorithm that is able to reconstruct the event sequences for the entire net that are consistent with: 1) the asynchronous observations of state changes; 2) the event sequences of each subnet; and 3) the structure of the given Petri net. We discuss the algorithmic complexity.

The final problem studied in this thesis is the optimal design method of Petri net controllers with fault-tolerant ability. In particular, we consider *multiple* faults detection and identification in Petri nets that have state machine structures (i.e., every transition in the net has only one input place and one output place). We develop the approximation algorithms to design the fault-tolerant Petri net controller which achieves the minimal number of connections with the original controller. A design example for an automated guided vehicle (AGV) system is also provided to illustrate our approaches.

1. INTRODUCTION

1.1 Background and Motivation

1.1.1 Background

Many systems, particularly technological ones, are in fact discrete-state systems [1]. Even if this is not the case, for many applications of interest a discrete-state view of a complex system may be necessary. The drawbacks of *continuous-variable dynamic systems* (CVDS), which are famous for continuous states and time-driven dynamics, in representing the above systems, encourage the development of *discrete event dynamic systems* (DEDS). To be more general, we call them *discrete event systems* (DESs). In contrast to CVDS, discrete states and event-driven dynamics are the most important features of DESs. We give the standard definition of DESs as follows.

Definition 1.1.1 *[1] A* Discrete Event System *(DES) is a* discrete-state, event-driven system*, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time.*

In this thesis, we focus on a particular DES model, i.e., Petri nets. Petri nets are a graphical and mathematical modeling tool applicable to many systems [2]. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems. Petri nets can be used by both practitioners and theoreticians. They are widely applied in many practical areas, such as intelligent transportation systems (ITS), sensor networks, and communication networks due to their event-driven feature and advantages in dealing with asynchronous processes.

Traffic system modeling and controlling is attracting more and more research attentions recently because of the increasing frequency of traffic congestions and accidents. Traffic systems are characterized by the continuous competition among vehicles for the occupation of certain physical zones in or around the intersections, which requires careful synchronization. Such problem is further complicated by the different physical layouts, vehicle-flow rates, turning movements, and pedestrian movements of each intersection.

Due to the scale and complexity of traffic systems, the dedicated models and control methods for them are necessary. Because of the synchronization requirement for resource sharing and the event-driven feature of the traffic light systems, Petri nets are powerful tools to solve traffic system modeling and controlling problems. In literature, Petri nets were used to build car safety controller model in road tunnels [3], construct safeness-enforcing supervisory control systems for railway networks [4], and build interaction model in intelligent vehicle control systems [5].

One of the important application areas of Petri nets in ITS is the control and deadlock avoidance in automated guided vehicles systems (AGVs). The authors of [6] implemented the automated manufacturing system using AGVs and modeled it by colored resource-oriented Petri net. Based on the model, the M-policy with polynomial complexity for deadlock avoidance was proposed. In [7], the authors transformed the simultaneous dispatching and routing problems for AGVs to the optimal transition firing problem in timed Petri nets and applied Petri net decomposition approaches to reduce the computational complexity. In [8], the AGV layout and paths were modeled by colored timed Petri nets and a deadlock avoidance strategy was proposed based on the results of the simulations.

Petri nets are used to model sensor networks and verify communication algorithms, too. Some properties of Petri nets, such as reachability, safeness, liveness, are important issues in this research area. With encoding of the Petri net state, we can also enable fault detection and identification capability in sensor networks and mitigate potential human errors.

In [9], the authors used space time Petri net (STPN) to model temporal and spatial information of sensor networks and simulate different behaviors. In [10], the authors utilized e-Petri net to provide suitable programming platform, seamless networking, data management, and service/resource discovery in wireless sensor networks (WSNs). In [11], the authors employed an augmented Petri net formalism called Extended Elementary Object System to model sensor networks. A synchronous firing mechanism was utilized as a security measure to detect malicious node attacks to sensor data and information flow. In [12], the authors proposed a decentralized Petri net based wireless sensor node architecture (PN-WSNA) which realized a reconfigurable WSN architecture and offered an easy graphical editing environment for constructing intelligent agents. In [13], the authors used Petri net to specify an attack-driven model in wireless sensor networks and verified their attack-resistant and efficient localization scheme when considering distance enlargement attacks. Through reachability analysis, they proved that the potential insecure states are unreachable. The authors in [14] developed a probabilistic model based on Petri nets to minimize energy consumption in wireless sensor networks. Their model showed advantages to Markov model in experiments. In [15], the author used the Petri net to model the operated behaviors in semiautonomous mobile sensor networks (MSNs) and utilized the supervisory control of Petri nets to implement the command filters, which effectively eliminated the undesirable executions. An interesting example of mobile wireless surveillance system involving mobile robots, MSNs, and command filters was provided in [15] to illustrate the above method.

In the aforementioned models based on Petri nets, the controllers synthesized were typically assumed to be fault-free. However, due to the complex operation dynamics, faults (such as sensor error and/or hardware interference) could occur at any time, which might cause the controller state to be corrupted and the control functionality to be significantly compromised. For this reason, fault-tolerance has received considerable attention in Petri net models. The basic rules for fault-tolerant controller design in Petri nets usually leave much flexibility for the choice of controller parameters. Then how to devise those parameters to achieve some optimization purposes directly affects the cost of implementing such fault-tolerant controllers. Plenty of research has been carried out on this topic like [16], [17], [18], and [19].

1.1.2 Motivation

After reviewing the preliminaries and standard notations of Petri nets in Chapter 2, we first study the modeling issue of traffic systems via Petri nets in Chapter 3. Some existed models describe the traffic systems thorough macroscopic indices, such as density, average speed and flow rate. Although the above strategies simplify the modeling process of traffic systems, they lose the functionality to guide individual drivers. Their common-used tools are continuous Petri nets and hybrid Petri nets, which require more complicated control schemes than the supervisory control strategies for discrete Petri nets. In contrast, there are several microscopic models which treat each vehicle as individual token. This feature is very important in AGVs. However, these microscopic models are not dedicated enough and ignore some practical issues. Besides the aforementioned problems, both kinds of models do not put enough attention on the traffic light system, which is the most common measure to regulate urban traffic flows. If we make the best of existed infrastructure, we can minimize the cost to implement the control strategies corresponding to our models. In this thesis, we focus on the microscopic modeling of intersections through timed discrete Petri nets and upgrade the traffic light system as another layer of the Petri net representation.

In Chapter 4, we study the event sequence reconstruction for sensor networks based on asynchronous observations. Consider the situation where sensors are used to gather information about state change of a sensor network and report their observations to a centralized observer. The absence of a global clock in the overall system implies that the centralized observer collects asynchronous information. However, some subsets of sensors may share some local clocks due to their similarity in construction or their short physical distances. This is one of the reasons that we introduce local observers to the original Petri net. As we will see later in Chapter 4, by adding the counting places, we can reduce the complexity of the algorithm to some extent compared to that of the algorithm proposed in [20]. The application prospect of our algorithm is quite promising. For instance, ITS is full of sensors to gather necessary data for the online adjustment of its control strategies. To ensure the fault-tolerant ability of its sensor system, it also has malfunction monitoring system. When several faults are detected together, we always hope to figure out which one is the source for the knock-on effect of faults.

Finally, we study the optimal design method of fault-tolerant controllers in systems modeled by Petri nets in Chapter 5. To guarantee the steady operation of controllers under possible faults, redundancy is necessary but cost increases at the same time. The optimization in this area is to minimize cost increment while keeping the same reliability and never interfering the original normal operation of controllers. Such redundancy is usually obtained through adding additional places and corresponding arcs to the original controller. The number of places added is determined by the number of faults we want to detect and identify simultaneously, which is a system specifications determined in advance.

Recall that in Petri net models, each arc (connection) corresponds to an nonzero entry in the input (or output) incident matrix of the net. In practice, each connection between the fault-tolerant controller and the plant is implemented by a programmable device and/or a circuit [21]. Assuming that each connection has the same hardware cost to be implemented, the problem of minimizing the overall hardware cost reduces to the problem of minimizing the total number of connections between the redundant controller and the original controller, and then to the problem of minimizing the total number of nonzero entries in the input and output incident matrices of the redundant controller. Such optimization criterion is more practical than that in [19], because in most contemporary controller implementation methods, the change of the value of arc weight (not the change from zero to nonzero or the change from nonzero to zero) does not affect the controller hardware implementation cost.

Before further discussing our approaches for modeling, monitoring and optimization of DESs, we state in more detail related work on these subjects and point out the similarities and differences with our research.

1.2 Related Work

1.2.1 Traffic System Modeling Based on Petri Nets

A variety of models based on Petri nets have been proposed for traffic systems. In terms of whether each vehicle in the traffic system is accurately described, these models are classified into two categories: macroscopic models and microscopic models.

In macroscopic traffic system models, continuous Petri nets and hybrid Petri nets (HPN) are usually used to describe the three key macroscopic parameters of the traffic systems, i.e., density, average speed and flow rate. Based on the infinite servers semantics, the discrete time model of continuous Petri net, and the finite-time emptying rule of places, the authors of [22] proposed a modular representation of road section through continuous Petri net, which served as building block in the complete traffic system model synthesis. Model predictive control (MPC) was applied to such model as verification in [23] with the purpose to minimize the total time delay of vehicles in the traffic system. Although such modular representations accurately described the capacity, inflow, outflow, and flow limitation of road sections, they are complicated when integrated to form a intersection model. A HPN model for urban traffic control was provided in [24]. Such model is composed of the subnets for the phase of traffic light, the incoming direction and the road section. The model of a T-shape intersection was used as example in [24].

Microscopic traffic system models usually utilize discrete Petri nets to simulate the detailed behavior of vehicles when they are approaching and crossing the intersections. Such dedicated models are able to lend vehicle drivers more guidance in or around the urban intersections. In recent years, more and more automatic manufacturing factories come out, which are famous for automatic material handling systems composed of AGVs. In AGV traffic systems, accurate guidance to cross the intersections is necessary for AGV's, which indicates the necessity of microscopic traffic system models. In [25], [8], and [19], AGV traffic systems were modeled by Petri nets and controlled through supervisory control strategies.

In [26], an ordinary Petri net model of a four-way intersection with a two-phase traffic light is proposed. The physical zones of the intersection is divided into four crossing sections (represented by four places). The availability of these crossing sections is controlled by corresponding controller places. However, the connection between crossing section places and the transferring transitions of these places suffers some synchronization problems. Moreover, the traffic light system subnet does not consider the time-driven characteristic.

In [27], the authors put forward a deterministic-timed Petri net model for the traffic system with multi-stage traffic light system. Such model is subsequently modified in [28], [29], and [30]. The authors of [31] improved such model. They considered different movements separately. They treated the interval times of vehicles in the traffic network, the time to travel a crossing section, and the cycle time of traffic light stage as stochastic values. However, one drawback of the model in [31] is that no left turn is allowed in the above model, which limits its application prospect. Moreover, some circular waiting deadlocks of the model in [31] is solved "autonomously" through some auxiliary Petri nets which is not helpful for the research of vehicular behaviors.

1.2.2 Transition Firing Sequence Reconstruction in Petri Nets

Using Petri nets to model and analyze sensor networks is a common method like the previous research results from [9] to [15] that we mentioned in Section 1.1.1. The problem of transition firing sequence reconstruction in Petri nets is also well-studied. However, most of existed reconstruction algorithms for transition firing sequence are conducted in labeled Petri nets, where each transition is assigned an observable or unobservable label. Such algorithms mainly deal with the languages generated by the labeled Petri nets.

In [20], the authors addressed the problem of reconstructing the transition firing sequences of a given Petri net based on asynchronous observations of token changes at different places of the Petri net. More specifically, they assumed that there exists a set of local sensors, each of which provided information about the token changes at a particular place of the Petri net. Having received information regarding the ordering of token changes at various places in the Petri net, the task of their algorithm was to reconstruct all possible transition firing sequences that were consistent with all sequences of token changes observed *and* the structure of the Petri net. The local sensors did not share any global timing information and were unaware of transitions that fired without affecting the tokens at their corresponding place. Therefore, the observed sequences of token changes only provided *partial* information about the order in which the number of tokens at different places changed. In this thesis, we consider the similar problem setup to that in [20] but develop a more efficient algorithm. Our initial result is presented in [32].

1.2.3 Optimization of Fault-Tolerant Controllers for Petri Net Models

The design of fault-tolerant controllers for Petri net models needs to ensure that the normal operation of original controllers are not interfered and that a certain number of faults (designated by control specifications in advance) can be detected and identified at the same time. In [16], the authors developed approaches for the design of redundant Petri net controllers with fault tolerance capabilities. The authors also provided the necessary and sufficient conditions for the design of such controllers but no optimal design criteria were discussed. In [17], the authors considered the similar setting in [16] and developed an algorithm to minimize the initial state of fault-tolerant Petri net controller. In [18], the authors considered another optimization criterion to minimize the sum of arc weights of the (input and output) incident matrices of the redundant controller and proposed an partial-order tree approach. However, they only considered the problem of single fault detection and identification. The authors of [19] put forward an approximation algorithm to minimize of the sum of arc weights of the incident matrices of the redundant controller with multiple faults detection and identification capability.

1.3 Major Contributions

Petri nets are powerful graphical and mathematical tools to model, analyze, and control large-scale systems, especially those display event-driven and asynchronous properties. Petri nets have received wide application in many practical fields such as ITS, sensor networks, and automatic manufacturing. As a result, the deep research on Petri nets lends us the insight to better understanding the complicated operation dynamics of various practical systems.

This thesis studies three important problems in Petri net models, i.e., traffic system modeling, transition firing sequence reconstruction and optimization of fault-tolerant controller. Our research in this thesis covers the complete process of applying Petri nets to solve practical problems, namely, modeling, monitoring, and optimization. Moreover,our solutions to the above problems are systematic. For traffic system modeling, we focus on modular representations of fundamental components. For transition firing sequence reconstruction, we divide the graphical solution into the steps of algorithms. For optimization of fault-tolerant controller, we conduct strict mathematical deduction and conclude the algorithm to achieve optimal purpose. Our goals are three-folds: 1) Design microscopic model of signalized intersection based on timed Petri nets to describe all kinds of vehicular behaviors and avoid deadlocks. 2) Develop decentralized algorithm to make the best of sensor network structure and solve transition firing sequence reconstruction problem with higher efficiency. 3) Propose the optimal design method of fault-tolerant controllers of Petri net models which minimized the number of connections between the original controller and the redundant part. Below is the detailed statement of our approaches and contributions to the above goals.

1.3.1 Signalized Intersection Modeling Through Timed Petri Nets

We propose a two-layer timed Petri net model for the signalized intersection in the microscopic sense. The first layer is the representation of the intersection which involves both deterministic-timed and stochastic-timed transitions. The second layer is the representation of the traffic light system which involves deterministic-timed and immediate transitions. Due to the more detailed division of the traffic light system cycle, our model allows all the three kinds of turning behaviors: going straight, turning left, and turning right. The different control policy in the yellow cycle (when the traffic light is yellow) compared to that of the green cycle solves the circular waiting deadlocks mentioned in [31].

1.3.2 Decentralized Algorithm for Transition Firing Sequence Reconstruction in Petri Nets

We consider the similar problem setup to that in [20] but develop a more efficient algorithm. More specifically, besides the set of asynchronously observed token change sequences, we assume that we have some local synchronous information. We first divide the original Petri net into several subnets. For each subnet, we add a local observer to the net which is called the *counting place* (which will be introduced in Chapter 4). Through the observed token change sequence of the counting place, we can reconstruct the transition firing sequence of each subnet. Then we develop an algorithm that is able to reconstruct the event sequences for the entire net that are consistent with: 1) the asynchronous observations of state changes; 2) the event sequences of each subnet; and 3) the structure of the given Petri net. We also discuss the algorithmic complexity and present an example to illustrate our approach.

1.3.3 Optimal Fault-Tolerant Controllers with Least Number of Connections for Petri Net Models

We consider the minimization of the number of arcs (the number of nonzero entries in the input and output incident matrices) of the redundant controller, rather than the minimization of the sum of the arc weights of redundant controllers in [19]. With the help of Reed-Solomon coding [33], we are able to develop an approximation algorithm to design the fault-tolerant Petri net controller in a systematic manner. A design example for an AGV system is also provided to illustrate our approach.

1.4 Organization

This thesis is organized as follows. After reviewing the standard notations and preliminaries of Petri nets in Chapter 2, we explain the timed Petri net model for the signalized intersection in Chapter 3. The characteristics and modeling requirements of the signalized intersection are stated at first. Then the corresponding Petri net representations to satisfy the above requirements is presented. In Chapter 4, we study the event sequence reconstruction in sensor networks modeled by Petri nets. Following some fundamental definitions, we formulate the problem. Then the algorithm to solve such problem is proposed and its complexity is analyzed. An example extracted from [20] is given to show our improvement. In Chapter 5, we put forward the approximation algorithm to optimize the structure of fault-tolerant controllers of Petri net models. After conducting detailed mathematical deduction, we give the approximation algorithm and prove its correctness. An example of AGV system is utilized to illustrate the procedure of such algorithm. We conclude this thesis and list the future research directions related to this thesis in Chapter 6.

2. NOTATION AND PRELIMINARIES

2.1 Introduction

Petri nets, due to their event-driven characteristic, are good at dealing with asynchronous process. It can clearly display the precedence relation among events. Moreover, the graphical representation of Petri nets are tightly related to mathematical operations, especially to the linear algebra theory and probability theory. Such relation makes Petri nets suitable for both practitioners and researchers, and emphasizes the importance to build necessary mathematical foundations before conducting deep research on Petri nets.

This section provides some basic definitions and terminology that will be used throughout the thesis. In Section 2.2, the graphical definitions and corresponding mathematical representations of Petri nets are given. The transition enabling condition and evolution pattern of Petri nets are also presented. The definition and transition enabling condition of timed Petri nets (deterministic and stochastic) are presented in Section 2.3. Some mathematical definitions that are useful in the deduction of Chapter 5 is listed in Section 2.4. We summarize our presentation in this chapter in Section 2.5. More details about Petri nets can be found in [2] and [1].

2.2 Petri Nets

A Petri net structure is a directed weighted bipartite graph $N = (P, T, A, W)$ where $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of *places* (drawn as circles), $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of *(immediate) transitions* (drawn as black bars), $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (from places to transitions and from transitions to places), and $W : A \to \{1, 2, 3, \ldots\}$ is the *weight function* on the arcs.

A *marking* is a vector $M : P \to \mathcal{N}$ (in what follows, $\mathcal{N} = \{0, 1, 2, \ldots\}$ denotes the set of nonnegative integer numbers) that assigns to each place in the Petri net a nonnegative integer number of tokens (drawn as black dots). We say a place is *l-bounded* ($l \in \{1, 2, 3, \ldots\}$) if there are at most l tokens in this place. We use $M[0]$ to denote the initial marking of the Petri net.

We use ${}^\bullet p$ (${}^\bullet t$) to denote the set of input transitions (places) of a place p (transition t) and $p^\bullet$ ($t^\bullet$) to denote the set of output transitions (places) of a place p (transition t). Let b_{ij}^- denote the integer weight of the arc from place p_i to transition t_j, and b_{ij}^+ denote the integer weight of the arc from transition t_j to place p_i ($1 \leq i \leq n$, $1 \leq j \leq m$). Note that b_{ij}^- (b_{ij}^+) is taken to be zero if there is no arc from place p_i to transition t_j (or vice versa). We define the *input incident matrix* $B^- = [b_{ij}^-]$ (respectively the *output incident matrix* $B^+ = [b_{ij}^+]$) to be the $n \times m$ matrix with b_{ij}^- (respectively b_{ij}^+) at its i-th row, j-th column position. The *incident matrix* of the Petri net is defined to be $B \equiv B^+ - B^-$. The state (or marking) evolution of Petri net is given by

$$M[k+1] = M[k] + (B^+ - B^-)x[k] \equiv M[k] + Bx[k], \tag{2.1}$$

where $M[k]$ is the marking of the Petri net at time epoch k, and $x[k]$ is the *firing vector* that is restricted to have exactly one nonzero entry with value "1," (when the j-th entry is "1," transition t_j fires at time epoch k). Note that transition t_j is *enabled* at time epoch k if and only if $M[k] \geq B^-(:, j)$, where the inequality is taken element-wise and $B^-(:, j)$ denotes the j-th column of B^-.

Let $\sigma = t_{i_1} t_{i_2} \ldots t_{i_k}$ ($t_{i_j} \in T$) be a transition firing sequence. We say σ is enabled with respect to M if $M[t_{i_1}\rangle M_1[t_{i_2}\rangle \ldots M_{k-1}[t_{i_k}\rangle$ where $M_j \geq 0$ ($j \in \{1, 2, \ldots, k-1\}$) denote a set of intermediate markings; this is denoted by $M[\sigma\rangle$. Let $M[\sigma\rangle M'$ denote that the firing of σ from M yields M' and let $\overline{\sigma}(t)$ be the total number of occurrences of transition t in σ. More specifically, $\overline{\sigma} = [\overline{\sigma}(t_1), \overline{\sigma}(t_2), \ldots, \overline{\sigma}(t_m)]^T$ is the *characteristic vector* that corresponds to σ. Note that after firing an enabled sequence σ from marking M, the new marking M' can also be computed as $M' = M + B\overline{\sigma}$.

2.3 Timed Petri Nets

If time delays associated with transitions and/or places are introduced, then we obtain timed Petri nets. If the delays are deterministic, such a Petri net model is called a *deterministic-timed Petri net.* If the delays follow some probability distribution, such a Petri net model is called a *stochastic-timed Petri net.* In this thesis, we only consider time delays that are associated with transitions and that describe the time from the enabling to the firing of transitions. The deterministic-timed transitions are illustrated by the white boxes while the stochastic-timed transitions are illustrated by the black boxes. In contrast, we assume that the immediate transitions (represented by the black bars) fire instantly once they are enabled.

Suppose the delay d_j, associated with transition t_j, is a nonnegative continuous random variable X with the exponential distribution function

$$F_X(x) = Pr[X \leq x] = 1 - e^{-\lambda_j x}, \tag{2.2}$$

and the probability density function

$$f_X(x) = \lambda_j e^{-\lambda_j x}. \tag{2.3}$$

Then, the average delay of t_j is

$$\overline{d_j} = \int_0^{\infty} [1 - F_X(x)]dx = \int_0^{\infty} e^{-\lambda_j x} dx = \frac{1}{\lambda_j}, \tag{2.4}$$

where λ_j is the firing rate of t_j.

Observation 2.3.1 *In a case where several timed transitions (deterministic or stochastic) are simultaneously enabled, the transition that has the shortest delay will fire first.*

2.4 Inequalities and Absolute Values of Matrix and Vector

Let $\mathcal{L}$ be the set of integer numbers. Given matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ in $\mathcal{L}^{n\times m}$, A (respectively B) is said to be nonnegative if $A \geq 0$ (respectively $B \geq 0$), i.e., if $a_{ij} \geq 0$ (respectively $b_{ij} \geq 0$) for every $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$.

Define $A \geq B$ if $a_{ij} \geq b_{ij}$ for every $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$; $A \leq B$ is defined in a similar way. The absolute value of matrix A, denoted by $|A|$, is to replace every element in A with its absolute value. That is, $|A| = [|a_{ij}|]$.

2.5 Summary

In this chapter, we reviewed some basics of Petri nets, such as the graphical representations and corresponding mathematical meanings, the markings, the transition enabling condition, and the evolution pattern. The difference between Petri nets and timed Petri nets was also discussed. Some important mathematical definitions were also presented. This chapter builds consolidate foundation for the following research in this thesis. In the next chapter, we will address the first problem of this thesis, namely, the traffic system modeling based on Petri nets.

3. MODELING OF SIGNALIZED INTERSECTION BASED ON TIMED PETRI NETS

3.1 Introduction

In this chapter, we propose a two-layer timed Petri net model for the signalized intersection in the microscopic sense. The first layer represents the intersection while the second layer represents the traffic light system. Timed transitions are involved in both of the two layers. The time delays of deterministic-timed transitions in the first layer corresponds to the time vehicles spend to cross certain physical zones in the intersection. The average time delays of stochastic-timed transitions in the first layer are used to distinguish vehicles with different turning behaviors. The time delays of deterministic-timed transitions in the second layer determine the duration of each light stage. There are also immediate transitions in the second layer to represents the shift of light stages. Our division strategy of the traffic light system cycle is dedicated enough to allow all the three kinds of turning behaviors: going straight, turning left, and turning right. The control policy in the yellow cycle (when the traffic light is yellow) solves the circular waiting deadlocks mentioned in [31]. Our initial results are presented in [34].

The organization of this chapter is as follows. In Section 3.2, we state the characteristics and functionalities of the signalized intersection model. In Section 3.2, the two-layer Petri net representation is provided which simulates the characteristics and functionalities stated in Section 3.3. The operation process of such Petri net model is also described in this section. We conclude this chapter in Section 3.4.

3.2 Urban Intersection Model

In this section, we first introduce the physical zone division of the intersection. Then we illustrate the physical zone occupation situation of vehicles under different movement behaviors (left turn, right turn and going straight). Based on the aforementioned structure, the traffic light system with phase division is proposed to rule all kinds of turning behaviors.

3.2.1 Division and Usage of Intersection

In this chapter, a signalized four-way intersection in the urban traffic network is divided into four crossing sections A, B, C, and D in counterclockwise order in order to clearly model vehicular behaviors when the vehicle crosses the intersection (see Fig. 3.1). Every crossing section can hold at most one vehicle at any time. A vehicle coming from left side (approaching section A) and going up (exiting from section C) crosses at first section A, then section B, and finally section C. As a result, to accomplish a left turn, a vehicle needs to cross three sections. A vehicle coming from left side (approaching section A) and going down (exiting from section A) crosses only zone A. To finish a right turn, a vehicle only needs to cross one section. A vehicle coming from left side (approaching section A) and going to right (exiting from section B) crosses at first section A and then section B. To go straight, a vehicle needs to cross two sections. Similar analysis can be applied to vehicles coming from down side (approaching section B), right side (approaching section C), and up side (approaching section D).

Observation 3.2.1 *In what follows, when we say "the vehicles approaching section A", we always mean the vehicles coming from the left side of the intersection and trying to across the intersection. It is similar when we say "the vehicles approaching section B", "the vehicles approaching section C" and "the vehicles approaching section D".*

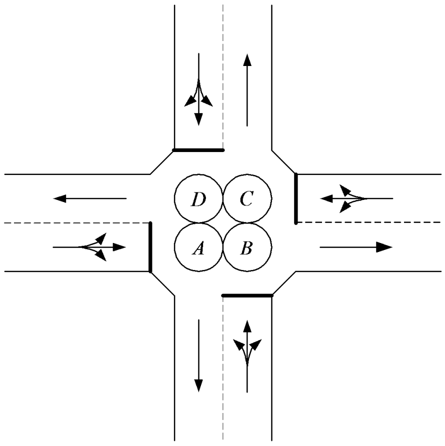

Fig. 3.1. An intersection with four crossing sections

3.2.2 Traffic Light System

The traffic light system cycle for the signalized urban intersection is designed to have two phases. In phase 1, the vehicles approaching sections A and C are allowed to cross the intersection while the vehicles approaching sections B and D are stopped from entering the intersection. In phase 2, the vehicles approaching sections B and D are allowed to cross the intersection while the vehicles approaching sections A and C are stopped from entering the intersection.

Each phase is again divided into two stages: left turn stage (L stage) and going straight or right turn stage (SR stage). L stage proceeds SR stage in a phase. When L stage begins (for example, the L stage of phase 1), the left turns of the vehicles approaching section A are allowed while the left turns of the vehicles approaching section C are prohibited. No going straight or right turn behavior is allowed. The vehicles approaching sections B and D are stopped from entering the intersection for sure. This is called the *green cycle* of section A of the L stage of phase 1. When the green cycle of section A of the L stage of phase 1 ends, the *yellow cycle* of of section A of the L stage of phase 1 begins. No vehicle is allowed to enter the intersection during this period. This period is used for vehicles staying in the intersection to finish their left turns. When the yellow cycle of section A of the L stage of phase 1 ends, another pair of green and yellow cycles follows which is designated to the vehicles approaching section C. After that, the L stage of phase 1 ends.

Observation 3.2.2 *The reason for the separation between the left turn from section A and the left turn from section C is as follows. The left turn from section A will occupy sections A, B and C in turn. The left turn from section C will occupy sections C, D and A in turn. Because of the common usage of sections A and C, such two kinds of left turn behaviors exclude each other. If one vehicle enters section A and tries to turn left while another vehicle enters section C at the same time and also tries to turn left, the traffic system will arrive in a circular wait deadlock. As a result, the separation of two kinds of left turn behaviors effectively avoids such deadlock. The above strategy is also widely used in urban traffic light systems.*

When the L stage of phase 1 ends, the SR stage of phase 1 begins. The *green cycle* of the SR stage of phase 1 starts at first. In this period, the vehicles approaching sections A and C can enter the intersection one by one and go straight or turn right freely. Notice that in this stage, the going straight behaviors and right turn behaviors of the vehicles approaching section A do not interfere the going straight behaviors and right turn behaviors of the vehicles approaching section C and vice versa. When the green cycle of the SR stage of phase 1 ends, the *yellow cycle* of the SR stage of phase 1 begins. No vehicles is allowed to enter the intersection. This period is used for the vehicles staying in the intersection to accomplish their going straight behaviors or right turn behaviors.

After the yellow cycle of the SR stage of phase 1 expires, phase 1 ends and phase 2 starts. The operation of phase 2 is similar to that of phase 1. After phase 2 ends, one whole cycle of the traffic light system ends and the next cycle begins.

Observation 3.2.3 *The design of the yellow cycles in both the L stage and the SR stage is utilized to eliminate the circular wait deadlock mentioned in [31] without the help of the deadlock-recovery stochastic-timed Petri nets. For example, the yellow* *cycle of the SR stage in the end of phase 1 leaves enough time for vehicles staying in the intersection and originally approaching sections A and C to accomplish their going straight behaviors or right turn behaviors. During this period, no new vehicle are allowed to enter the intersection. As a result, when phase 2 begins, there is no vehicle staying in the intersection. We can conclude that the vehicles for different phases never affect each other. Based on the similar analysis, we can even conclude that the vehicles for different cycles and different stages never affect each other. Hence, the situation mentioned in [31], where four vehicles originally approaching the four crossing sections and trying to go straight occupy the four sections at the same time and therefore lead to a deadlock, can never happen under the design of this chapter.*

3.3 Petri Net Representation

The two-layer Petri net representation of the above signalized intersection model is provided in this section with detailed definitions of each place and transition. We first show the Petri net representation of the intersection with the four crossing sections. Then we propose the Petri net representation for the traffic light system, which regulations the operation of the Petri net representation of the intersection. We finally focus on the representation subnet of one crossing section A and explain the interaction between the above two layers of Petri net representations.

To better understand the Petri net representations, we introduce the notation rules for the places and transitions as follows. The capital letters "A", "B", "C", and "D" represent the four crossing sections A, B, C, and D. The capital letter "L" means "left turn", the capital letter "S" means "going straight", and the capital letter "R" means "right turn". The lower case letter "q" represents "queue". The lower case letter "a" means "availability". The lower case letters "oc" means "output controller". The lower case letter "b" means "begin" while the lower case letter "e" means "end". The lower case letter "g" represents "green" and the lower case letter "y" represents "yellow". The "$\rightarrow$" represents the transfer of the states.

The double arrow arc between a place and a transition means that this place is both the input place and the output place of the transition. Moreover, the input arc weight and the output arc weight are the same. Recall that the immediate transition is represented by the black bar, that the deterministic-timed transition is represented by the white box, and that the stochastic-timed transition is represented by the black box.

3.3.1 Petri Net Representation of Intersection

We display the Petri net representation of the intersection in Fig. 3.2. In places p_{Aa}, p_{Ba}, p_{Ca}, and p_{Da}, a token means the availability of the corresponding crossing section. In places p_{Aoc}, p_{Aoc}, p_{Aoc}, and p_{Aoc}, a token means that the corresponding crossing section allow exiting. The above places are all 1-bounded. In other places, tokens represent single vehicles. Because the operation of the Petri net representation of the intersection is regulated by the Petri net representation of the traffic light system, we will analyze the Petri net representation of the intersection in details after we introduced the Petri net representation of the traffic light system. We provide the definitions of the places and transitions of the Petri net representation of the intersection in Table 3.1 and Table 3.2 respectively.

3.3.2 Petri Net Representation of Traffic Light System

The Petri net representation of the traffic light system is shown in Fig. 3.3. All places in Fig. 3.3 are 1-bounded. It interacts with the Petri net representation of the intersection in Fig. 3.2 through places p_{AL}, p_{CL}, p_{ASR}, p_{CSR}, p_{BL}, p_{DL}, p_{BSR}, and p_{DSR}. For instance, p_{AL} is both the input place and the output place of transition t_{AL} in the Petri net representation of the intersection. t_{AL} is enabled only if there is one token in p_{AL}. Other places interacts with the corresponding transitions in the Petri net representation of the intersection in similar ways.

Table 3.1 The definitions of the places in Fig.3.2

Places	Definitions
p_{Xq} $(X \in \{A, B, C, D\})$	Each token in p_{Xq} represents a vehicle approaching section X and trying to cross intersection.
p_{XLq} $(X \in \{A, B, C, D\})$	Each token in p_{XLq} represents a vehicle approaching section X and trying to turn left at intersection.
p_{XSRq} $(X \in \{A, B, C, D\})$	Each token in p_{XSRq} represents a vehicle approaching section X and trying to go straight or turn right at intersection.
p_{Xa} $(X \in \{A, B, C, D\})$	A token in p_{Xa} means section X is available and no vehicle occupies this section. No token in p_{Xa} means section X is not available.
p_{Xout} $(X \in \{A, B, C, D\})$	Each token in p_{Xout} represents a vehicle ready to exit from the intersection through section X.
p_{Xoc} $(X \in \{A, B, C, D\})$	A token in p_{Xoc} means section X allows exiting. No token in p_{Xoc} means section X does not allow exiting.

Table 3.2 The definitions of the transitions in Fig.3.2

Transitions	Definitions
t_{Xin} $(X \in \{A, B, C, D\})$	Firing of t_{Xin} simulates the flow of vehicles approaching section X.
t_{XLin} and t_{XSRin} $(X \in \{A, B, C, D\})$	Firing of t_{XLin} and t_{XSRin} divides vehicles approaching section X into those turning left and those going straight or turning right.
t_{XS} and t_{XR} $(X \in \{A, B, C, D\})$	Firing of t_{XS} and t_{XR} specifies vehicles approaching section X and going straight and vehicles approaching section X and turning right. When t_{XS} or t_{XR} fires, a vehicle enters intersection through section X.
t_{XL} $(X \in \{A, B, C, D\})$	Time delay of t_{XL} represents the extra time that a vehicle approaching section X spends turning left compared to the time that a vehicle approaching section X spends going straight or turning right. When t_{XL} fires, a vehicle enters intersection through section X.
t_{Xout} $(X \in \{A, B, C, D\})$	Firing of t_{Xout} represents a vehicle exits from intersection through section X. Time delay of t_{Xout} represents the time that a vehicle takes to cross intersection.

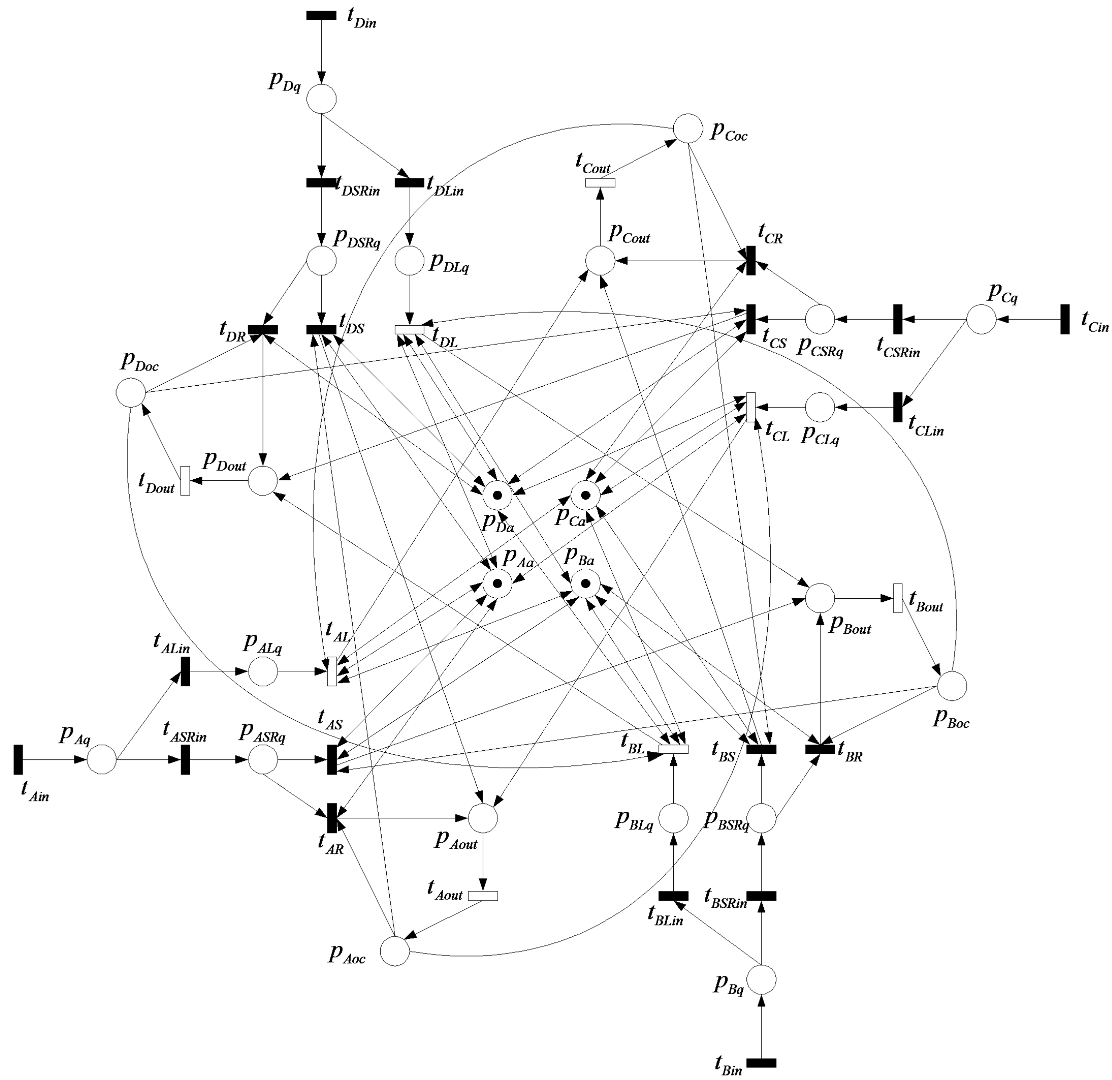

Fig. 3.2. The Petri net representation of the intersection of Fig. 3.1

Each deterministic-timed transition (the white box) in Fig. 3.3 represents a state of the traffic light system. When there is one token in its input place (notice that every deterministic-timed transition in Fig. 3.3 has only one input place), it means the traffic light system stays in the state corresponding to such deterministic-timed transition (such transition is also enabled). The time delay of such deterministic-timed transition defines how long the traffic light system stays in the corresponding state. Each deterministic-timed transition also has only one output place. When such transition fires and there is one token in its output place, it means the duration of the corresponding state of such transition expires. Each immediate transition (the black bar) represents the transfer between the corresponding states. The definitions of the places and transitions in Fig. 3.3 are listed in Table 3.3 and Table 3.4.

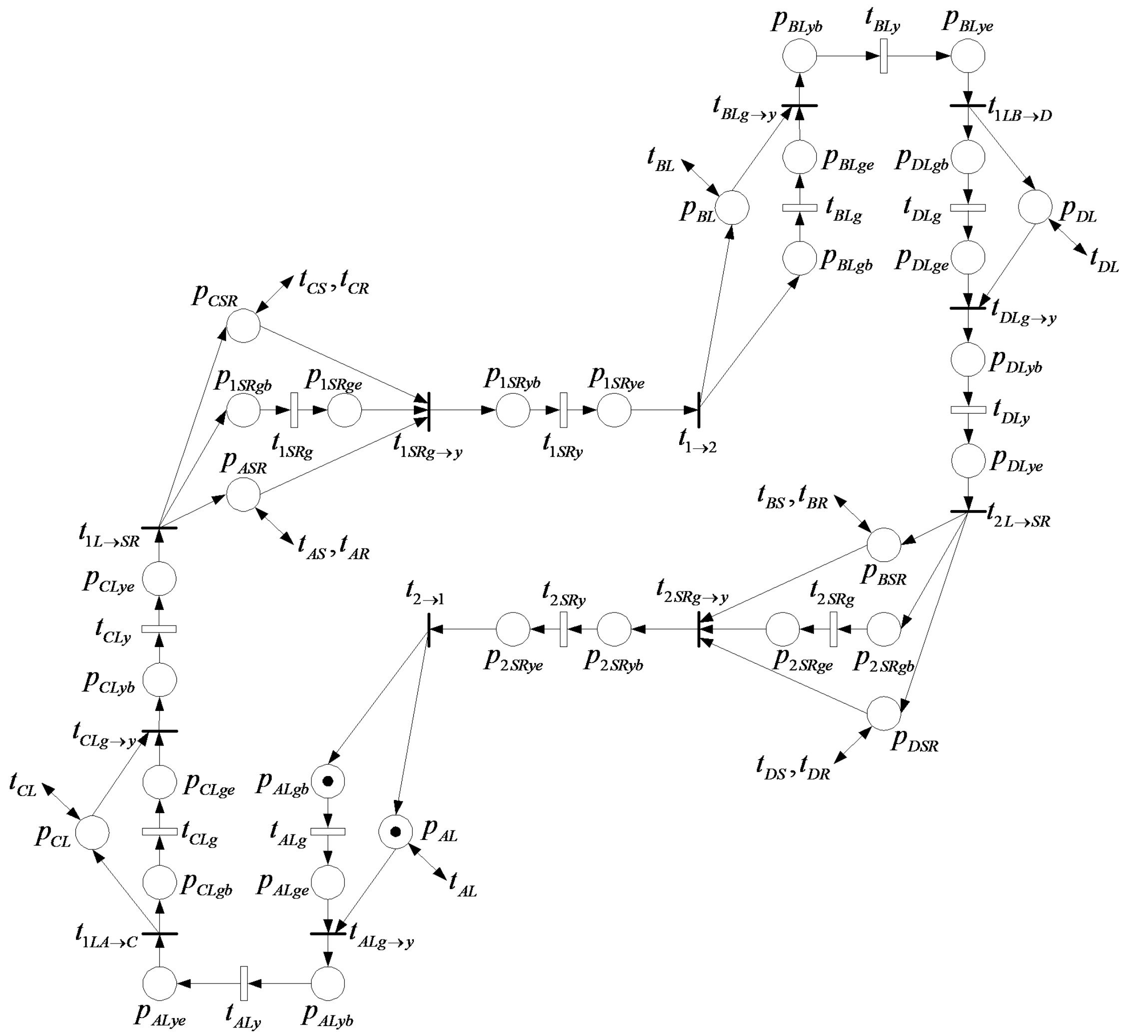

Fig. 3.3. The Petri net representation of the traffic light system

Observation 3.3.1 *In Fig. 3.3, one token is deposited in both places p_{ASR} and p_{CSR} when transition $t_{1L\to SR}$ fires. p_{ASR} interacts with transitions t_{AS} and t_{AR} while p_{CSR} interacts with transitions t_{CS} and t_{CR}. Such separation satisfies the requirement that*

Table 3.3 The definitions of the places in Fig.3.3

Places	Definitions
p_{XLgb} $(X \in \{A, B, C, D\})$	A token in p_{XLgb} means that green cycle of section X in corresponding L stage begins.
p_{XLge} $(X \in \{A, B, C, D\})$	A token in p_{XLge} means that green cycle of section X in corresponding L stage ends.
p_{XL} $(X \in \{A, B, C, D\})$	A token in p_{XL} means that vehicles approaching section X can turn left.
p_{XLyb} $(X \in \{A, B, C, D\})$	A token in p_{XLyb} means that yellow cycle of section X in corresponding L stage begins.
p_{XLye} $(X \in \{A, B, C, D\})$	A token in p_{XLye} means that yellow cycle of section X in corresponding L stage ends.
p_{xSRgb} $(x \in \{1, 2\})$	A token in p_{xSRgb} means that green cycle of SR stage of phase x begins.
p_{xSRge} $(x \in \{1, 2\})$	A token in p_{xSRge} means that green cycle of SR stage of phase x ends.
p_{XSR} $(X \in \{A, B, C, D\})$	A token in p_{XSR} means that vehicles approaching section X can go straight or turn right.
p_{xSRyb} $(x \in \{1, 2\})$	A token in p_{xSRyb} means that yellow cycle of SR stage of phase x begins.
p_{xSRye} $(x \in \{1, 2\})$	A token in p_{xSRye} means that yellow cycle of SR stage of phase x ends.

Table 3.4 The definitions of the transitions in Fig.3.3

Transitions	Definitions
$t_{x\to x'}$ ($x, x' \in \{1,2\}$ and $x \neq x'$)	Firing of $t_{x\to x'}$ represents phase x ends and phase x' begins.
t_{XLg} ($X \in \{A,B,C,D\}$)	Time delay of t_{XLg} represents duration of green cycle of section X in corresponding L stage.
$t_{XLg\to y}$ ($X \in \{A,B,C,D\}$)	Firing of $t_{XLg\to y}$ represents green cycle of section X in corresponding L stage ends and yellow cycle of section X in corresponding L stage begins.
t_{XLy} ($X \in \{A,B,C,D\}$)	Time delay of t_{XLy} represents duration of yellow cycle of section X in corresponding L stage.
$t_{xLX\to X'}$ ($x = 1$, $X = A$, and $X' = C$, or $x = 2$, $X = B$, and $X' = D$)	Firing of $t_{xLX\to X'}$ represents yellow cycle of section X in corresponding stage ends and green cycle of section X' in corresponding stage begins.
$t_{xL\to SR}$ ($x \in \{1,2\}$)	Firing of $t_{xL\to SR}$ represents L stage of phase x ends and SR stage of phase x begins.
t_{xSRg} ($x \in \{1,2\}$)	Time delay of t_{xSRg} represents duration of green cycle of SR stage of phase x.
$t_{xSRg\to y}$ ($x \in \{1,2\}$)	Firing of $t_{xSRg\to y}$ represents green cycle of SR stage of phase x ends and yellow cycle of SR stage of phase x begins.
t_{xSRy} ($x \in \{1,2\}$)	Time delay of t_{xSRy} represents duration of yellow cycle of SR stage of phase x.

in the SR stage of phase 1, the going straight and right turn behaviors of the vehicles approaching section A do not interfere the going straight or right turn behaviors of the vehicles approaching section C and vice versa. Places p_{BSR} and p_{DSR} work in the similar way. This is an significant improvement compared to the traffic light system controlling rule in [31]. In [31], vehicles approaching different crossing sections which belong to the same phase need to compete for the traffic light system resource in the SR stage, which is not realistic.

3.3.3 Cooperation of Places and Transitions in Petri Net Representation

The definitions of the places and transitions in the Petri net representation of the signalized urban intersection are listed in Section 3.3.1 and Section 3.3.2. Based on the above definitions, the cooperation process of those places and transitions to simulate and regulate the traffic flow across the intersection will be stated in details in this section. Due to the symmetry of the four crossing sections, we will focus on the representation subnet of section A and the corresponding transitions and places in the Petri net representation of the traffic light system. The representation subnets of other sections and their corresponding transitions and places in the Petri net representation of the traffic light system operate similarly. We abstract the places, transitions and arcs related to the subnet of section A from Fig. 3.2 and show them in Fig. 3.4.

In Fig. 3.4, transition t_{Ain} is a stochastic-timed transition with the exponential distribution. t_{Ain} models the vehicle approaching process for section A. All the vehicles (tokens) approaching section A are queued (contained) in place p_{Aq}.

Transitions t_{ALin} and t_{ASRin} are two stochastic-timed transitions with very small time delay since they are used for decision-making to divide the vehicles approaching section A into those turning left and those going straight or turning right. The ratio of average time delays between t_{ALin} and t_{ASRin} equals the ratio between the percentage of vehicles approaching section A and turning left and the percentage of vehicles approaching section A and going straight or turning right. Such decision-making

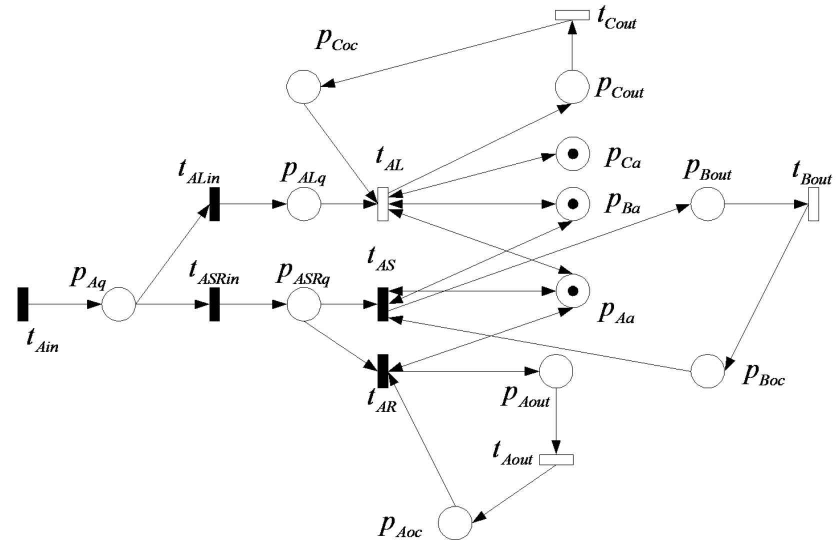

Fig. 3.4. The representation subnet of the crossing section A of Fig. 3.1

process should be very fast compared to the intersection crossing process of vehicles. This idea is enlightened by the decision-making transitions in the conflict-solved small stochastic-timed Petri net in [31]. Such two kinds of vehicles (tokens) are queued (contained) in places p_{ALq} and p_{ASRq} respectively. Similarly, transitions t_{AS} and t_{AR} are two stochastic-timed transitions with very small time delay. The ratio of average time delays between t_{AS} and t_{AR} equals the ratio between the percentage of vehicles approaching section A and going straight and the percentage of vehicles approaching section A and turning right. t_{AS} and t_{AR} distinguish the vehicles approaching section A and going straight from those approaching section A and turning right.

In contrast, transition t_{AL} is a deterministic-timed transition. The time delay of t_{AL} equals the average extra time that a vehicle approaching section A spends turning left compared to the time that a vehicle approaching section A spends going straight or turning right. The reason of such extra time cost for left turn is that a vehicle needs to cross three crossing sections to finish a left turn and vehicles always slow down when turning. We do not distinguish between the time that a vehicle spends going straight and the time that a vehicle spends turning right. Although a vehicle only crosses one crossing section when turning right compared to two crossing sections for going straight, the speed of a vehicle when it goes straight is higher than its speed when it turns right.

When each of places p_{Aa}, p_{Ba}, and p_{Ca} contains one token (sections A, B, and C are available), there is one token in place p_{Coc} (vehicles are allowed to exit from the intersection through section C), and there is one token in place p_{AL} of Fig. 3.3 (which means the traffic light system stays in the green cycle for vehicles approaching section A of the L stage of phase 1 based on the analysis in Section 3.3.2), then t_{AL} is enabled. When the time delay of t_{AL} expires, a token is removed from p_{Aq} and is deposited in p_{Cout} (there is also one token removed from p_{Coc}, which stops other vehicles approaching section A from turning left). Then deterministic-timed transition t_{Cout} is enabled. When the time delay of t_{Cout} expires, a token is removed from p_{Cout} (a vehicle finishes the left turn and exits from the intersection through section C) and is deposited in p_{Coc} (other vehicles approaching section A are allowed to turn left now). The time that a vehicle approaching section A spends finishing a left turn is the summation of the time delays of t_{AL} and t_{Cout}.

The enabling and firing process of transitions t_{AS} and t_{AR} are similar to the enabling and firing process of t_{AL}. However, t_{AS} and t_{AR} may be enabled at the same time and therefore in the conflict for the token in p_{Aa}, p_{ASR} of Fig. 3.3, and p_{ASRq} (if there is only one token in p_{ASRq}). The above three places are the common input places for t_{AS} and t_{AR}. Such conflict is solved by the different average time delays between t_{AS} and t_{AR}. The transition with less average time delay is more likely to fire. Recall that the ratio of the average time delays between t_{AS} and t_{AR} equals to the ratio between the percentage of vehicles approaching section A and going straight and the percentage of vehicles approaching section A and turning right. The time a vehicle approaching section A spends finishing a right turn is the time delay of t_{Aout} while the time a vehicle approaching section A spends finishing a going straight behavior is the time delay of t_{Bout}.

Observation 3.3.2 *The reason why we queue the vehicles (tokens) approaching section A and turning left in p_{ALq} separately from the vehicles (tokens) approaching section A and going straight or turning right is as follows. t_{AL} is enabled and fires under the green cycle for section A of the L stage of phase 1, prior to the SR stage of phase 1 when t_{AS} and t_{AR} are enabled and fire. If we delete t_{ASRin}, t_{ALin}, p_{ASRq}, and p_{ALq}, change t_{AL} into the stochastic-timed transition (the ratio of the average time delays among t_{AL}, t_{AS}, and t_{AR} equals the ratio of the percentages of vehicles approaching section A and taking corresponding turning behaviors), and set t_{AL}, t_{AS}, and t_{AR} as the output transitions of p_{Aq}, then the firing times of t_{AL} may be much larger than the real number of vehicles approaching section A and turning left since t_{AL} does not have "competitors" during the green cycle for section A of the L stage of phase 1. As a result, we separate the vehicles approaching section A and turning left from the vehicles approaching section A and going straight or turning right one step earlier (in t_{ALin} and t_{ASRin}) than when we separate the vehicles approaching section A and going straight from the vehicles approaching section A and turning right (in t_{AR} and t_{AS}). We do not split t_{ASRin} into t_{ASin} and t_{ARin} or split p_{ASRq} into p_{ASq} and p_{ARq} due to the competition between t_{AS} and t_{AR} under the same state of the traffic light system.*

From Fig. 3.1 and Fig. 3.2, we see that section A is the exit of the right turn vehicles from section A, the going straight vehicles from section D and the left turn vehicles from section C. Since the time that a vehicle approaching section C spends finishing a left turn can be adjusted by the time delay of t_{CL}, the time delay of t_{Aout} should be the average between the time that a vehicle approaching section A spends finishing a right turn and the time that a vehicle approaching section D spends finishing a going straight behavior. Based on similar analysis, we can deduct the formulas for the time delays of transitions t_{Bout}, t_{Cout}, and t_{Dout}.

Observation 3.3.3 *We integrate the output controller places p_{Aout}, p_{Bout}, p_{Cout}, and p_{Dout} into the Petri net representation of the intersection of Fig. 3.2 due to the following reason. (We will focus our analysis on section A.) t_{Ain} simulates the vehicles* *approaching process for section A so that its time delay is physically meaningful. The time delays of decision-making transitions* t_{AS} *and* t_{AR} *are extremely short compared to that of* t_{Ain}. *As a result, we need* t_{Aout} *and* t_{Bout} *with physically meaningful time delays to match the time delay of* t_{Ain} *and simulate the going straight and right turn behaviors of vehicles approaching section A. Without the control of* p_{Aout} *and* p_{Bout}, *vehicles (tokens) approaching section A and going straight or turning right gather together in* p_{Aout} *and* p_{Bout} *even under the rule of the traffic light system since the time delays of* t_{AS} *and* t_{AR} *are extremely short. In the real world, this corresponds to the situation that many vehicles crowd in the intersection and lead to traffic jam. With the control of* p_{Aout} *and* p_{Bout}, *vehicles approaching section A and going straight or turning right can only cross the intersection one by one.*

3.4 Summary

In this chapter, a two-layer timed Petri net model was proposed for the signalized intersection in the microscopic sense. The first layer was the representation of the intersection and the second layer was the representation of the traffic light system. We stated the definitions of places and transitions in the above two Petri net representations. Based on these definitions, we described the cooperation process between the two Petri net representations to simulate and regulate the vehicle flow across the signalized intersection. The improvements of such model compared to the previous models were also discussed. In the next chapter, we will discuss the algorithm to reconstruct transition firing sequences in Petri net models.

4. SENSOR NETWORK MONITORING BASED ON ASYNCHRONOUS OBSERVATIONS

4.1 Introduction

In this chapter, we consider the similar problem setup to that in [20] but develop a more efficient algorithm. More specifically, besides the set of asynchronously observed token change sequences, we utilize some local synchronous information. We divide the original Petri net into several subnets at first. For each subnet, we add a local observer to it, which is called the *counting place* and will be introduced in the next section. Through the observed token change sequence of the counting place, we can reconstruct the transition firing sequence of each subnet. Then we develop an algorithm that is able to reconstruct the event sequences for the entire net that are consistent with: 1) the asynchronous observations of state changes; 2) the event sequences of each subnet; and 3) the structure of the given Petri net. We also discuss the algorithmic complexity and present an example to illustrate our approach. As we will see later in this chapter, by adding the counting places, we can reduce the complexity of the algorithm to some extent compared to the one proposed in [20].

The remainder of this chapter is organized as follows. In Section 4.2, some basic definitions are given with several illustrative examples. In Section 4.3, we formulate our problem to be studied. We propose our reconstruction algorithm and analyze its complexity in Section 4.4. An example is also provided for illustration and comparison with the algorithm proposed in [20]. We conclude this chapter in Section 4.5.

4.2 Basic Definitions

We first list some definitions specific to this chapter. These definitions are quite useful when we state the transition firing sequence reconstruction problem in Section 4.3 and when we explain the reconstruction algorithm procedure in Section 4.4. Several illustrative examples accompany these definitions.

Definition 4.2.1 *Let $\mathcal{S}^n$ denote the space of sequences of markings in $(Z^+)^n$ and define $\Gamma_A : \mathcal{S}^n \to \mathcal{S}^A$ be the* projection *that focuses on the sequence of marking changes at places indexed by the set A* and *also removes repeated elements in the sequence.*

For example, the projection Γ_{p_i} of the sequence of markings $M[0], M[1], \ldots, M[k]$ at the i-th place is given by

$$\begin{aligned}
&\Gamma_{p_i}(M[0] \to M[1] \to \ldots \to M[j_1-1] \to M[j_1] \to M[j_1+1] \to \ldots \\
&\quad \to M[j_k] \to \ldots \to M[k]) \\
=&M(p_i)[0] \to M(p_i)[j_1] \to M(p_i)[j_2] \to \ldots \to M(p_i)[j_k],
\end{aligned}$$

where $\{j_1, j_2, \ldots j_k\}$ is exactly the set of time epochs at which the number of tokens in place p_i changes. More specifically,
$\{j_1, j_2, \ldots j_k\}$ satisfies

$$\begin{aligned}
M(p_i)[0] = M(p_i)[1] = \ldots = M(p_i)[j_1-1] &\neq M(p_i)[j_1], \\
M(p_i)[j_1] = M(p_i)[j_1+1] = \ldots = M(p_i)[j_2-1] &\neq M(p_i)[j_2], \\
&\vdots
\end{aligned}$$

Consider a sequence of markings in $(Z^+)^3$ that is given by

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 3 \\ 2 \end{bmatrix}.$$

If $A = \{1, 3\}$,

$$\Gamma_A \left(\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 3 \\ 2 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$

Assume that each sensor is responsible for reporting token changes in one place in the net. We denote the observed token change sequence at each place p_i ($i \in \{1, 2, \ldots, n\}$) by $s_i = M(p_i)[0] \rightarrow M(p_i)[j_{i_1}] \rightarrow M(p_i)[j_{i_2}] \rightarrow \ldots \rightarrow M(p_i)[j_{i_k}]$ (note that $0 < j_{i_1} < j_{i_2} < \ldots < j_{i_k}$). Let $M(p_i)[0]$ ($M(p_i)[j_{i_k}]$) denote the initial (final) number of tokens in place p_i. We use S to denote the set of all observed sequences of token changes, i.e., $S = \{s_1, s_2, \ldots, s_n\}$.

Definition 4.2.2 *[20] Let $\sigma = t_{i1}t_{i2}\ldots t_{ik}$ be a transition firing sequence such that $M_0[t_{i1}\rangle M_1[t_{i2}\rangle \ldots [t_{ik}\rangle M_k$, where $M_j \geq 0$ ($j \in \{1, 2, \ldots, k\}$) denote a sequence of markings in the net. We say σ is a* consistent transition firing sequence *with respect to the observed sequence s_i of token changes at place p_i if it satisfies $\Gamma_{p_i}(M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow \ldots \rightarrow M_k) = s_i$. Similarly, given the set of observed sequences of token changes $S = \{s_1, s_2 \ldots s_n\}$, σ is said to be a* consistent transition firing sequence with respect to S *if it is consistent with each sequence s_i ($i \in \{1, 2, \ldots, n\}$).*

For each sequence s_i ($i \in \{1, 2, \ldots, n\}$), let $I_i = \{0, 1, 2, \ldots, |s_i| - 1\}$ be the set of possible *position indices* for sequence s_i. The indexing process assigns to each element in s_i (from left to right) a unique nonnegative integer in the set $\{0, 1, 2, \ldots, |s_i| - 1\}$, in increasing order from left to right, to denote its position in the sequence ($|s_i|$ is the length of the observed sequence s_i and the leftmost element in the sequence is assigned index 0). More specifically, for $k_i \in I_i$, $s_i[k_i]$ is the number of tokens at place p_i after k_i-th token changes at that place. We use $I = [I_1 \ I_2 \ \ldots \ I_n]$ to denote the n-dimensional *position index vector* that captures the position indices of all observed sequences in S.

Consider the Petri net shown in Fig. 4.1 with three places $\{p_1, p_2, p_3\}$, three transitions $\{t_1, t_2, t_3\}$, and the initial marking $M[0] = [3\ 0\ 0]^T$. Suppose the set of observed token change sequences $S = \{s_1, s_2, s_3\}$ are given by

$$s_1 : \quad 3 \to 1 \to 0,$$
$$s_2 : \quad 0 \to 1 \to 0,$$
$$s_3 : \quad 0 \to 1 \to 3.$$

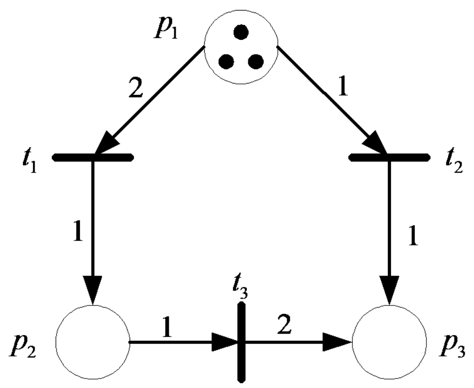

Fig. 4.1. A simple Petri net

In this case, it is not hard to show that $\{t_1 t_2 t_3\}$ is the only transition sequence that can possibly generate the observations in S. The marking evolution under $t_1 t_2 t_3$ is given by

$$\begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{t_1} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{t_2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{t_3} \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix},$$

and one can verify that the evolution of the number of tokens at each place (after projection Γ_{p_i} for each place p_i) satisfies

$$\Gamma_{p_1}(M[0] \to M[1] \to M[2] \to M[3]) = s_1,$$
$$\Gamma_{p_2}(M[0] \to M[1] \to M[2] \to M[3]) = s_2,$$
$$\Gamma_{p_3}(M[0] \to M[1] \to M[2] \to M[3]) = s_3.$$

Note that the evolution of the *position index vector* I that is associated with each consistent marking is given by

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \xrightarrow{t_1} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \xrightarrow{t_2} \begin{bmatrix} 2 & 1 & 1 \end{bmatrix} \xrightarrow{t_3} \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}.$$

In [20], the authors proposed a centralized transition firing sequence reconstruction algorithm without any synchronous information. In this thesis, we consider the situation where the given net have some locally synchronous information. This property is achieved by adding the *counting place* to the net, which is similar to the concept of observer in the setting of labeled Petri nets. We first partition the Petri net N into several subnets $N_1, N_2, \ldots, N_q$ where q is the number of subnets. Such partition is in terms of the transition set T. The corresponding transition sets for $N_1, N_2, \ldots, N_q$ are $T_1, T_2, \ldots, T_q$ respectively. The place set P is not affected by this partition. Some transitions can belong to more than one subnet. Such transitions are called the *common transition* of these subnets. Then we add to each subnet $N_i (i = 1, 2, \ldots, q)$ a counting place p_{ci}. Each transition in N_i has an *output* arc to the counting place p_{ci}. The weights of such output arcs in N_i are all *distinct*. Thus, when we obtain the observed token change sequence s_{ci} from p_{ci}, we can deduce which transition in N_i has fired according to the token increment at each time epoch. Following this approach, we are able to find the *sub transition firing sequence* F_i of each subnet N_i. We use the following example to illustrate the construction process of the counting places.

The Petri net N shown in Fig. 4.2 is extracted from Example 4 in [20]. In Fig. 4.3, N is partitioned into two subnets N_p and N_q, where the partition is illustrated by the dotted line in Fig. 3. N_p includes the transitions t_1, t_3, and t_5. The transitions t_2, t_4, and t_5 are contained in N_q. We construct the counting place p_{cp} (which is illustrated by the bold circle) for N_p. The (dashed) output arc weights from t_1, t_3, and t_5 to p_{cp} are 1, 2, and 3, respectively. Similarly, we construct p_{cq} for N_q and the (dashed) output arc weights from t_2, t_4, and t_5 to p_{cq} are 1, 2, and 3, respectively. Note that t_5 is the common transition between N_p and N_q.

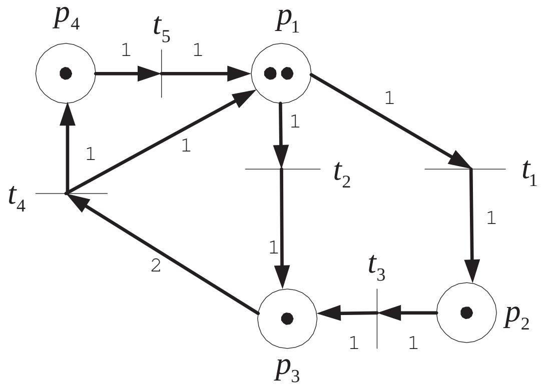

Fig. 4.2. Petri net N extracted from Example 4 in [20]

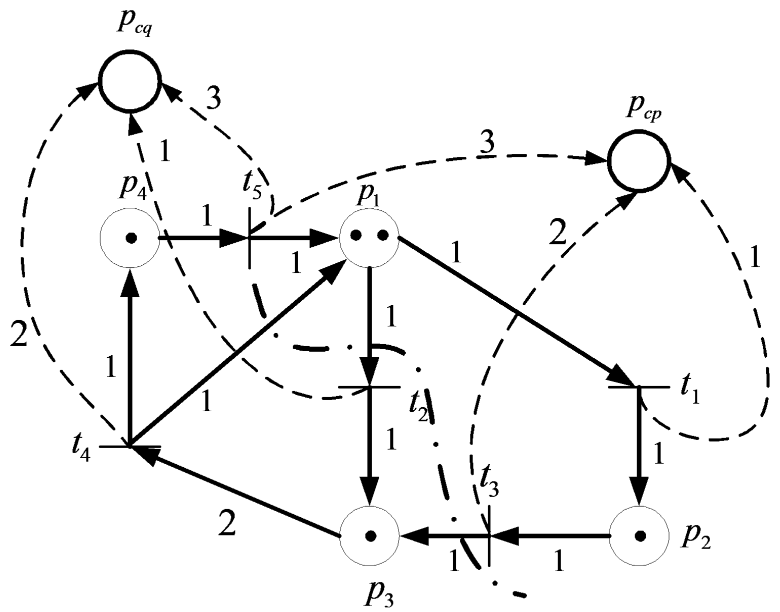

Fig. 4.3. Construction of counting places for Petri net shown in Fig 4.2

4.3 Problem Formulation

The problem we deal with in this thesis is as follows. We are given a Petri net N with initial marking $M[0]$, a set of n observed token change sequences $S = [s_1 \ s_2 \ \ldots \ s_n]$ that are provided asynchronously by local sensors for *each* of the n

places of the net, and a partition of N into two subnets N_p and N_q [1]. Our goal is to find all possible transition firing sequences that are consistent with both S and the structure of N. We assume that at each time epoch only one transition may fire. We also assume that the firing of each transition in the net changes the number of tokens in at least one of its input/output places.

Notice that the set of observed token change sequences S is provided asynchronously by local sensors; this implies that we only have *partial* information about the transition sequences that have fired (and the consistent markings associated with them). Moreover, in order to reconstruct the transition firing sequences, we need to capture the evolution of actual markings in the net based on S, the structure of the net, and the transition firing sequences of N_p and N_q which is provided by the *counting places*. Given the (asynchronously observed) set of token change sequences $S = \{s_1, s_2, \ldots, s_n\}$ for each place in the net, the difficulty with this problem is how to determine the ordering with which different transitions have fired. One important observation to keep in mind is that the token changes of each place p_i ($i \in \{1, 2, \ldots, n\}$) take place exactly in the order given in S; in addition, the structure of the net provides information about transitions that can fire under a particular marking M; in addition, the progress of the transition firing sequence of N should be compatible with the progress of the transition firing sequences of N_p and N_q.

Definition 4.3.1 *[20] The* current *marking $M[k]$ (i.e., the marking at current time epoch k) is given by $M[k] = [S[k_i]] = [s_1[k_1]\ s_2[k_2]\ \ldots\ s_n[k_n]]^T$, where k_i denotes the index within s_i and $s_i[k_i]$ denotes the current marking of place p_i.*

In the remainder of this section we describe informally an algorithm that can be used to reconstruct all possible transition firing sequences given knowledge of the Petri net structure, the observation of sequences of token changes at different places in the net, and the transition firing sequences of the subnets. We will describe the algorithm more formally in the next section.

[1] This setup can be easily extended to cases where N is partitioned into multiple subnets.

We assume that the *counting place* for a subnet is connected to each transition belonging to this subnet through an output arc. The weights of these output arcs in a subnet are all distinct. As a result, if we obtain the observed token change sequence of a counting place, we can easily reconstruct the transition firing sequence for the corresponding subnet. Note that different subnets may share some common transitions. The firings of such common transitions are captured in different subnets. Hence, these common transitions play as the role of synchronization points among subnets.

The idea of our algorithm is as follows. Starting from the initial marking $M[0]$, we first find the set of transitions that can fire in the next step of each transition firing sequence of each subnet. Then, we figure out the set of transitions that are enabled under the current marking (which is initially $M[0]$). After that, we obtain the intersection of these two sets. We compute the markings after the firings of the transitions in this intersection set and check the consistency with the set of observed sequences of token changes S; for those markings that are consistent, we update the position index vector and store the marking information (re-defined as current marking) and the corresponding transition that has fired (in order to reach this current marking). We also record the progress of each transition firing sequence for each subnet. Then, we go on to find, for each marking information stored, the next possible firing transitions, by repeating the steps described above.

Observation 4.3.1 *When there are several valid transitions that can fire, the algorithm chooses one transition and keeps on searching with it; others are considered only after we finish searching with this transition, i.e., the algorithm searches valid transition firing sequences in depth-first fashion [35].*

Observation 4.3.2 *If the transition firing sequence of one subnet reaches a common transition while the transition firing sequence of the other subnet does not, then the first transition firing sequence will "wait" for the latter one until it reaches the same common transition.*

Each time the position index vector matches those of the last element in each s_i (to ensure that there are no further token changes); the algorithm returns a solution according to the information stored. Then it goes back to the previous (stored but not explored) transition to keep looking for other possible firing sequences. The algorithm stops after exploring all transition sequences that could lead us from the initial marking to the final marking. A breadth-first search version of the algorithm that can be modified for online applications is also possible but it is not presented here due to space limitations.

4.4 Transition Firing Sequence Reconstruction

4.4.1 Reconstruction Algorithm

In this section, we describe an iterative algorithm that recovers the transition firing sequences in depth-first fashion. In the algorithm, we suppose that the original Petri net N is divided into two subnets N_p and N_q. However, note that our algorithm here can be easily extended to the cases with more than two subnets.

We first introduce some variables that will be used in our algorithm. F_p (or respectively, F_q) denotes the transition firing sequence of N_p (or respectively, N_q). Define l_p (or respectively, l_q) as the length of F_p (or respectively, F_q). The index of F_p (or respectively, F_q) is from 0 to $l_p - 1$ (or respectively, $l_q - 1$). f_p (or respectively, f_q) denotes the current index of F_p (or respectively, F_q).

T_{pq} denotes the set of common transitions between N_p and N_q. Define X_p (or respectively, X_q) as the position (the index in F_p or F_q) sequence of the common transitions for N_p (or respectively, N_q). The length of X_p and X_q is the same and is denoted by l_x. The index of X_p and X_q is from 0 to $l_x - 1$. We add an element $X_p[l_x] = l_p$ (or respectively, $X_q[l_x] = l_q$) to the end of X_p (or respectively, X_q). $X_p[l_x]$ and $X_q[l_x]$ serve as sentinels. The current index of X_p and X_q is l. If there are more than one common transition between N_p and N_q, all of their positions will be included in X_p and X_q. The common transitions play as the *synchronization points* between F_p

and F_q and divide F_p and F_q into corresponding segments. We only need to combine the corresponding segments.

F_c denotes the combined transition firing sequence of N_p and N_q. The length of F_c is $l_c = l_p + l_q - l_x$. Define the current time epoch as k and the current index of F_c is just k. The index of F_c is from 0 to $l_c - 1$.

$I[k] = [k_1\ k_2\ \ldots\ k_n]$ denotes the position index vector associated with the current marking $M[k]$, where k is the current time epoch. $k_i (i = 1, 2, \ldots, n)$ is the current index of the observed token change sequence s_i and $M(p_i)[k] = s_i[k_i]$.

$T_f[k]$ denotes the set of transitions consistent with the progress of F_p and F_q at time epoch k. $T_e[k]$ denotes the set of enabled transitions consistent with the structure of the Petri net N under $M[k]$. $T_p[k] = T_f[k] \bigcap T_e[k]$ denotes the set of transitions that we will use to check whether they satisfy the process of the set of observer token change sequences S.

We use structure $\mathcal{C}[k] = \{M[k], t_{in}, M[k-1], I[k], l, f_p, f_q, T_p[k]\}$ to capture the information we need to store each time a new marking is explored, where the transition t_{in} is enabled under $M[k-1]$ at the previous time epoch $k-1$ such that $M[k-1][t_{in}\rangle M[k]$.

The transition firing sequence reconstruction algorithm (which we call Algorithm 1) is described as follows.

Algorithm 1 starts at the initial marking and finds possible firing transitions iteratively by checking the consistency of markings that are obtained after their firings with the transition firing sequences F_p and F_q of the subnets N_p and N_q (Line 7 to Line 15), the structure of the Petri net N (Line 16 to Line 17), and the set of observed sequences of token changes S (Line 18 to Line 19). We first choose the set $T_f[k]$ of transitions that are consistent with the progress of the two transition firing sequences F_p and F_q. If none of the progress of F_p and F_q arrive at a common transition, then the two current transitions of F_p and F_q will be included in $T_f[k]$. If F_p arrives at a common transition and F_q does not, only the current transition of F_q will be included into $T_f[k]$. It means that F_p will wait for F_q. The common transitions serve as the

Algorithm 1 Transition firing sequence reconstruction

Input

- Petri net N with input/output incident matrix B^{-}/B^{+} and initial marking $M[0]$
- A set of observed token change sequences $S = \{s_1, s_2, \ldots, s_n\}$
- The set of common transitions T_{pq} between subnets N_p and N_q
- The observed token change sequences s_p and s_q of counting place p_{cp} and p_{cq}

Construct the transition firing sequence F_p and F_q of the subnets N_p and N_q according to s_p and s_q, respectively

Construct the common transition position sequence X_p and X_q according to F_p, F_q and T_{pq}

Set the current index l of X_p and X_q, the current index f_p of F_p and the current index f_q of F_q all by 0

Set the current time epoch $k = 0$. Set the current indices of the observed token change sequences as $k_1 = k_2 = \cdots = k_n = 0$ and construct the position index vector $I[0]$ accordingly

Store $\mathcal{C}[0] = \{M[0], \phi, \phi, I[0], 0, 0, 0, \phi\}$

Let $M[k] = [s_1[k_1]\ s_2[k_2]\ \ldots\ s_n[k_n]]^T$

if $f_p < X_p[l]$ and $f_q < X_q[l]$ **then**

    $T_f[k] = \{F_p[f_p], F_q[f_q]\}$

else

    if $f_p == X_p[l]$ and $f_q < X_q[l]$ **then**

        $T_f[k] = \{F_q[f_q]\}$

    else

        $T_f[k] = \{F_p[f_p]\}$

if $f_p == X_p[l]$ and $f_q == X_q[l]$ **then**

    $l = l + 1$

$T_e[k] = \{t_j \in T \mid \mathcal{C}.M[k] \geq B^{-}(:, t_j)\}$

Store $\mathcal{C}[k].T_p[k] = T_f[k] \bigcap T_e[k]$

18: **if** There exists $t_j \in \mathcal{C}[k].T_p[k]$ that has not been explored **then**

19: Pick up t_j and mark t_j as explored in $\mathcal{C}[k].T_p[k]$

20: $M'[k] = \mathcal{C}.M[k] + B(:, t_j)$

21: Calculate $M''[k] = [s_1[k'_1]\ s_2[k'_2]\ \ldots\ s_n[k'_n]]^T$ according to S, where $k'_i = k_i + 1$

22: $(i = 1, 2, \ldots, n)$ for $p_i \in {}^\bullet t_j$ or $p_i \in t_j^\bullet$ with $b^+_{ij} - b^-_{ij} \neq 0$ and $k'_i = k_i$ for others

23: **if** $M'[k] \neq M''[k]$ **then**

24: **Goto Line 18**

25: **else**

26: Update $I[k+1]$ as $k_i = k'_i (i = 1, 2, \ldots, n)$

27: **if** $t_j == F_p[f_p]$ **then**

28: $f_p = f_p + 1$

29: **else**

30: $f_q = f_q + 1$

31: Store $\mathcal{C}[k+1] = \{M'[k], t_j, M[k], I[k+1], l, f_p, f_q, T_p[k]\}$

32: $F_c[k] = \mathcal{C}[k+1].t_{in}$

33: $k = k + 1$

34: **if** $f_p == X_p[l]$ and $f_q == X_q[l]$ and $l == l_x$ **then**

35: **Goto Line 39**

36: **else**

37: **Goto Line 7**

38: **else**

39: **Goto Line 41**

40: **if** $k == (l_p + l_q - l_x)$ **then**

41: Store F_c as a solution

42: $k = k - 1$

43: **if** $k == -1$ **then**

44: Done

45: **else**

46: Return to $\mathcal{C}[k]$ and **Goto Line 18** synchronization points between F_p and F_q. The situation for F_q arrives at a common transition and F_p does not is similar. When both F_p and F_q arrive at a common transition, such common transition will be included into $T_f[k]$. Then we construct the set $T_e[k]$ of transitions enabled under marking $M[k]$. The intersection of $T_f[k]$ and $T_e[k]$ consists of the transition set $T_p[k]$ that will be evaluated whether they are valid or not by checking consistency with S; if a transition is valid, the algorithm stores the necessary marking information and keeps searching for new possible firing transitions from this marking onwards. When we reach the final marking (given in S) and there are no remaining entries in S, the algorithm returns the corresponding transition firing sequence with the information stored; then, it goes back to search for other possible (stored but not unexplored) transition firing sequences.

Observation 4.4.1 *In Algorithm 1, Line 31 is to update the structure* $\mathcal{C}$*. Note that the component* $T_p[k]$ *in* $\mathcal{C}$ *has not been updated yet at the moment and it will be updated in the next time when we execute Step* 5*.*

Let us consider Fig. 4.3.We assume that the set of token change sequences $S = \{s_1, s_2, s_3, s_4\}$ are given by

$$
\begin{aligned}
s_1 &: \quad 2 \to 1 \to 2 \to 3 \to 2 \to 3,\\
s_2 &: \quad 1 \to 2 \to 1,\\
s_3 &: \quad 1 \to 2 \to 0 \to 1,\\
s_4 &: \quad 1 \to 2 \to 1 \to 0,\\
s_{cp} &: \quad 0 \to 1 \to 3 \to 6 \to 9,\\
s_{cq} &: \quad 0 \to 2 \to 5 \to 6 \to 9.
\end{aligned}
$$

s_{cp} and s_{cq} are the observed token change sequences of the counting places p_{cp} and p_{cq}, respectively. According to s_{cp} and s_{cq}, we can construct the transition firing sequences F_p and F_q of the subnets N_p and N_q, respectively. The common transition t_5 between N_p and N_q has been underlined.

$$F_p: \quad t_1 \rightarrow t_3 \rightarrow \underline{t_5} \rightarrow \underline{t_5},$$
$$F_q: \quad t_4 \rightarrow \underline{t_5} \rightarrow t_2 \rightarrow \underline{t_5}.$$

The initial marking is $M[0] = [2\ 1\ 1\ 1]^T$. Clearly, the set of transitions consistent with the progress of F_p and F_q is $T_f[0] = \{t_1, t_4\}$. The set of enabled transitions under initial marking are $T_e[0] = \{t_1, t_2, t_3, t_5\}$. $T_p[0] = T_f[0] \bigcap T_e[0] = \{t_1\}$. The firing of t_1 is consistent with S. Therefore, Algorithm 1 runs iteratively from t_1 in depth-first search fashion. We obtain $M[1] = [1\ 2\ 1\ 1]^T$ after firing t_1 from $M[0]$. The complete result is shown in Fig. 4.4, where ID captures the order of markings visited. The position index vector is also shown under each marking explored.

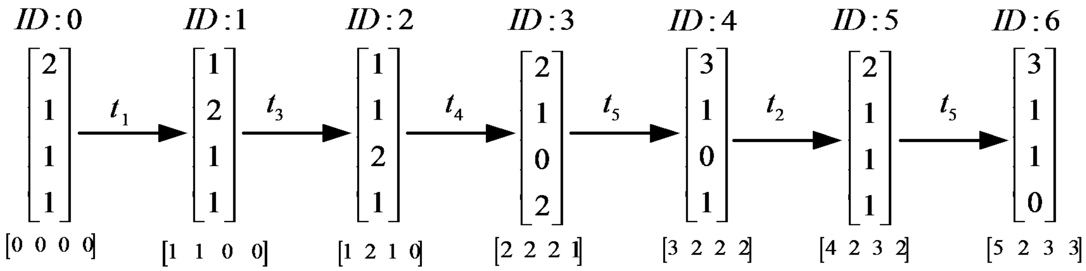

Fig. 4.4. Complete results for Fig. 4.3

Regarding the data structure stored, for marking $[1\ 2\ 1\ 1]^T$ (with $ID = 1$ as shown in Fig. 4.4), the information stored is given by $\mathcal{C}[1] = \{[1\,2\,1\,1]^T, t_1, [2\,1\,1\,1], [1\,1\,0\,0], 0, 1, 0, \{t_1\}\}$. Note that although the markings with $ID = 0$ and $ID = 5$ are identical, they behave differently because their position indices are different.

Finally, the set of possible transition firing sequences that are consistent with both S, the net N, F_p and F_q are given by:

$$\{\{t_1\ t_3\ t_4\ t_5\ t_2\ t_5\}\}.$$

Recall that Fig. 4.2 is extracted from [20]. We add counting places to the Petri net shown in Fig. 4.2 and obtain Fig. 4.3. The Petri net shown in Fig. 4.2 is also the algorithm illustration example in [20]. It takes more steps to finish the algorithm running in [20] compared to the steps of our algorithm running displayed in Fig. 4.4.

4.4.2 Complexity Analysis

In [20], the authors proved that the space complexity of their algorithm is $O(md)$ and the computational complexity is $O(m^d)$, where m is the number of transitions and d is the upper bound of the maximum length of the possible transition firing sequences.

For our algorithm, the space complexity is $O(2d) = O(d)$. The computational complexity is $O(2^d)$ because the maximum number of transitions we explore under each marking is 2 (due to the progress of F_p and F_q). Note that if the original Petri net N is partitioned into b subnets where $b > 2$, then the space complexity is $O(bd)$ and the computational complexity is $O(b^d)$ because the maximum number of transitions we explore for each marking is b.

4.5 Summary

In this chapter, we proposed a methodology for reconstructing possible transition firing sequences in a given Petri net based on asynchronous observations of the set of sequences of token changes in its places. We assumed that the observation of each marking change sequence was made by a local sensor and that there was no global timing (so that each sensor only knew the order of local marking changes). The original Petri net was partitioned into several subnets and the transition firing sequence of each subnet could be reconstructed through some special local observers. Based on the local observations from each sensor and each local observers, we developed an algorithm that was able to reconstruct all transition firing sequences that were consistent with these observations and the structure of the Petri net. The proposed algorithm proceeded in depth-first search fashion and iteratively reconstructed possible transition firing sequences. We also discussed the complexity of the algorithm and presented an example for illustration. We will state our optimization work on fault-tolerant controllers for Petri net models in the next chapter.

5. OPTIMIZATION OF FAULT-TOLERANT CONTROLLERS FOR PETRI NET MODELS

5.1 Introduction

In this chapter, we focus on multiple faults detection and identification, but with a completely different optimization criterion as proposed in [19]. More specifically, we consider the minimization of the number of arcs (the number of nonzero entries in the input and output incident matrices) of the redundant controller, rather than the minimization of the sum of the arc weights of redundant controllers in [19]. With the help of Reed-Solomon coding [33], we are able to develop an approximation algorithm to design the fault-tolerant Petri net controller in a systematic manner. A design example for an AGV system is also provided to illustrate our approach.

The organization of this chapter is as follows. In Section 5.2, we present the necessary and sufficient conditions to construct fault-tolerant controllers. The characteristics of such fault-tolerant controllers are also illustrated. We propose the optimization purpose in Section 5.3. Following the optimization purpose and the characteristics of fault-tolerant controllers, we deduct the approximation algorithm to achieve such optimization purpose. The correctness of such algorithm is also proved in this section. In Section 5.4, an example about AGV system is presented to show the procedure of the approximation algorithm. We conclude this chapter in Section 5.5.

5.2 Design of Fault-Tolerant Redundant Petri Net Controllers

In this section we briefly review the design approach for the fault-tolerant redundant Petri net controller and the procedure to perform fault detection and identification that were proposed in [16]. The optimal design of such redundant controller will be discussed in the next section.

5.2.1 Fault-Tolerant Redundant Controllers

In [16], the authors proposed approaches for the design of fault-tolerant Petri net controllers, which can be summarized as follows.

Let $\mathcal{C}$ be a given Petri net controller with n_c places, m transitions, and state evolution given by

$$M_c[k+1] = M_c[k] + \left(B_c^+ - B_c^-\right) x[k] \equiv M_c[k] + B_c x[k], \tag{5.1}$$

where $x[k]$ is the firing vector, B_c^+ is the output incident matrix of $\mathcal{C}$, B_c^- is the input incident matrix of $\mathcal{C}$, and $B_c \equiv B_c^+ - B_c^-$ is the incident matrix of $\mathcal{C}$.

Consider a larger Petri net $\mathcal{H}$ with $\eta = n_c + 2d$ $(d > 0)$ places[1], m transitions, and state evolution given by

$$M_h[k+1] = M_h[k] + \left(\mathcal{B}_c^+ - \mathcal{B}_c^-\right) x[k] \equiv M_h[k] + \mathcal{B}_c x[k], \tag{5.2}$$

where $x[k]$ is the firing vector, $\mathcal{B}_c^+$ is the output incident matrix of $\mathcal{H}$, $\mathcal{B}_c^-$ is the input incident matrix of $\mathcal{H}$, and $\mathcal{B}_c \equiv \mathcal{B}_c^+ - \mathcal{B}_c^-$ is the incident matrix of $\mathcal{H}$.

Proposition 5.2.1 *[16] A fault-tolerant redundant controller $\mathcal{H}$ is bisimulation equivalent to the given controller net $\mathcal{C}$ if and only if there exists matrices C and D satisfying conditions (C1) and (C2) as follows.*
(C1) $C \geq 0$ is a $2d \times n_c$ matrix with nonnegative integer entries and;
(C2) $D \geq 0$ is a $2d \times m$ matrix with nonnegative integer entries such that $D \leq \min\left(CB_c^+, CB_c^-\right)$, where the inequality taken element-wise.

Conditions (C1) and (C2) characterize necessary and sufficient conditions for the design of matrices C and D such that the fault-tolerant redundant controller $\mathcal{H}$ is bisimulation equivalent to the original controller $\mathcal{C}$ (i.e., any transition firing sequence enabled in the original controller is also enabled in the redundant one, and vice versa).

[1]It has been shown in [36] that by adding $2d$ places, we can detect and identify d place faults.

5.2.2 Fault Detection and Identification

In [16], the authors considered the detection and identification of *place faults*. Place faults at time epoch k results in an erroneous marking $M_f[k]$ that can be expressed as

$$M_f[k] = M_h[k] + e_p, \tag{5.3}$$

where $M_h[k]$ is the marking of the redundant controller that would have been reached under fault-free conditions, and e_p is the place error vector. A possibly erroneous marking $M_f[k]$ can be checked by using the parity check matrix

$$P = \begin{bmatrix} -C & I_{2d} \end{bmatrix}, \tag{5.4}$$

where I_{2d} is the $2d \times 2d$ identity matrix, to verify whether the syndrome, defined as

$$s[k] \equiv PM_f[k], \tag{5.5}$$

is equal to 0. Clearly, in the place fault model, the syndrome at time epoch k is given by

$$\begin{aligned} s[k] &\equiv PM_f[k] = P(M_h[k] + e_p) \\ &= P(GM_c[k] + e_p) = 0 + Pe_p = Pe_p, \end{aligned} \tag{5.6}$$

and fault detection and identification is exclusively determined by matrix P. We assume that the number of tokens in the redundant Petri net does not get affected by place faults, therefore, the syndrome in (5.6) is reduced to the form as follows.

$$s^*[k] = Ce_p^*, \tag{5.7}$$

where e_p^* contains the first n_c entries of e_p. To be able to detect and identify d place faults, we can choose matrix C such that any linear combination of d or less columns of matrix C is *unique*. In other words, matrix C should satisfy condition (C3) for multiple faults detection and identification given as follows.

(C3) The choice of C should guarantee that any linear combination of d or less columns of matrix C is unique.

In [36], the author used Reed-Solomon coding to design matrix C (matrix H in [36]), which satisfies condition (C3). C is in the form as follows.

$$C \equiv \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_{n_c} \\ \alpha_1^2 \bmod p & \alpha_2^2 \bmod p & \dots & \alpha_{n_c}^2 \bmod p \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{2d} \bmod p & \alpha_2^{2d} \bmod p & \dots & \alpha_{n_c}^{2d} \bmod p \end{bmatrix}, \tag{5.8}$$

where the prime $p > n_c$, $2d \leq n_c$ and $\alpha_1, \alpha_2, \dots, \alpha_{n_c}$ are n_c distinct nonzero elements in $GF(p)$.

Observation 5.2.1 *The columns of C are interchangeable. In Reed-Solomon coding, the first line in matrix C consists of n_c distinct nonzero elements in $GF(p)$ without certain order. Furthermore, we will see in the following proposition that we could only focus on the linear combination of the columns in C.*

Proposition 5.2.2 *[37] Let C be the matrix defined in (5.8) and assume the prime $p > n_c$ and $2d \leq n_c$. Then,*

(i) Any $2d$ columns are linearly independent;

(ii) Any linear combination of d or less columns is unique.

Proposition 5.2.3 *All the entries in the matrix C are nonzero.*

Proof We claim that if $q^k \bmod p \neq 0$ where $q \in GF(p)$ and $q \neq 0$, then $q^{k+1} \bmod p \neq 0$. We prove our claim by contradiction. Suppose $q^{k+1} \bmod p = 0$. According to Lemma 5 in Section 2 of [38], $q^k \bmod p = 0$ or $q \bmod p = 0$. However, $q^k \bmod p \neq 0$ and $q \bmod p \neq 0$ according to given conditions. As a result, $q^{k+1} \bmod p \neq 0$ and our claim is true.

We already know that the first row of C, i.e., $1, 2, \dots, n_c$ are nonzero elements in $GF(p)$. Then $1^i, 2^i, \dots, n_c^i (i = 1, 2, \dots, 2d)$ are all nonzero according to our claim. Hence, all the entries in C are nonzero. ■

Note that there are many choices of matrices C and D that satisfy conditions (C1), (C2), and (C3) given above. In the next section, we will discuss our approach to design these matrices in order to minimize the sum of arc weights of the (input and output) incident matrices of the redundant controller.

5.3 Algorithm Design for Fault-Tolerant Redundant Petri Net Controllers

5.3.1 Problem Formulation

The problem we deal with in this thesis is the following. Consider a Petri net controller $\mathcal{C}$ (with n_c places, m transitions, and state evolution given in Equation (5.1)), design matrices C and D such that the resulting fault-tolerant redundant Petri net controller $\mathcal{H}$ (with $\eta = n_c + 2d$ places, m transitions, and state evolution given in Equation (5.2)) satisfies:

(i) $\mathcal{H}$ is bisimultion equivalent to $\mathcal{C}$ (i.e., satisfies conditions (C1) and (C2));

(ii) $\mathcal{H}$ is able to detect and identify d place faults (i.e., satisfies condition (C3));

(iii) the number of nonzero entries in the input and output incident matrices of $\mathcal{H}$ is minimized.

We introduce the definition of the *zero identification function* $\mathrm{sgn}(x)$ here.

$$\mathrm{sgn}\,(x) = \begin{cases} 0, & \text{if } x = 0; \\ 1, & \text{if } x \neq 0. \end{cases} \tag{5.9}$$

Since the input incident matrix of $\mathcal{H}$ is an $\eta \times m$ matrix denoted by $\mathcal{B}_c^-$ and the output incident matrix of $\mathcal{H}$ is an $\eta \times m$ matrix denoted by $\mathcal{B}_c^+$, this problem can be formulated as an optimization problem as follows.

$$\arg\min_{C,D} \sum_{i=1}^{\eta} \sum_{j=1}^{m} \Big(\mathrm{sgn}(\left[\mathcal{B}_c^-\right]_{ij}) + \mathrm{sgn}(\left[\mathcal{B}_c^+\right]_{ij}) \Big), \tag{5.10}$$

such that matrices C and D satisfy conditions (C1), (C2), and (C3), where $[\mathcal{B}_c^-]_{ij}$ (respectively, $[\mathcal{B}_c^+]_{ij}$) denotes the entry at the i-th row, j-th column in matrix $\mathcal{B}_c^-$

(respectively, $\mathcal{B}_c^+$), i.e., the arc weight from the place p_i to the transition t_j (respectively, from the transition t_j to the place p_i). If $[\mathcal{B}_c^-]_{ij} = 0$, there is no arc from the place p_i to the transition t_j. If nonzero, there is such arc. Similarly, if $[\mathcal{B}_c^+]_{ij} = 0$, there is no arc from the transition t_j to the place p_i. If nonzero, there is such arc. The choices of C and D do not affect the solution of the problem in (5.10) for the first n_c rows but the rest $2d$ rows. Therefore, the problem in (5.10) can be reduced to (5.11) as follows.

$$\arg\min_{C,D} \sum_{i=1}^{2d} \sum_{j=1}^{m} \left(\operatorname{sgn}\left(\left[CB_c^+\right]_{ij} - D_{ij} \right) + \operatorname{sgn}\left(\left[CB_c^-\right]_{ij} - D_{ij} \right) \right), \tag{5.11}$$

such that matrices C and D satisfy conditions (C1), (C2), and (C3).

Note that condition (C2) requires that matrix D is a matrix with nonnegative integer entries such that $0 \leq D \leq \min(CB_c^+, CB_c^-)$. Similar as the proof in [18], we proved the following proposition.

Proposition 5.3.1 *The solution of (5.11) subject to condition (C2) corresponds to the choice of matrix D such that D satisfies $D_{ij} = \min\left([CB_c^+]_{ij}, [CB_c^-]_{ij}\right)$ for every $i \in \{1, 2, \ldots, 2d\}$ and $j \in \{1, 2, \ldots, m\}$.*

Proof We prove this proposition by contradiction. Suppose the matrix D satisfies $D_{ij} = \min\left([CB_c^+]_{ij}, [CB_c^-]_{ij}\right)$ for every $i \in \{1, 2, \ldots, 2d\}$ and $j \in \{1, 2, \ldots, m\}$ except a single entry D_{kl} $(0 \leq D_{kl} < \min([CB_c^-]_{kl}, [CB_c^+]_{kl}))$ at its k-th row, l-th column position.

$D_{kl} < \min([CB_c^-]_{kl}, [CB_c^+]_{kl})$ implies that $[CB_c^-]_{kl} - D_{kl} \neq 0$ and $[CB_c^+]_{kl} - D_{kl} \neq 0$. As a result, $\operatorname{sgn}([CB_c^-]_{kl} - D_{kl}) + \operatorname{sgn}([CB_c^+]_{kl} - D_{kl}) = 2$. Since one of $[CB_c^-]_{kl} - \min([CB_c^-]_{kl}, [CB_c^+]_{kl})$ and $[CB_c^+]_{kl} - \min([CB_c^-]_{kl}, [CB_c^+]_{kl})$ equals zero and the other is nonzero, $\operatorname{sgn}([CB_c^-]_{kl} - \min([CB_c^-]_{kl}, [CB_c^+]_{kl})) + \operatorname{sgn}([CB_c^+]_{kl} - \min([CB_c^-]_{kl}, [CB_c^+]_{kl})) = 1$. This implies that (5.11) is not minimized. Contradiction comes out. ∎

From Proposition 5.3.1, it is not difficult to show the following three cases.

1) If $[CB_c^+]_{ij} > [CB_c^-]_{ij}$, then we have $D_{ij} = [CB_c^-]_{ij}$, $\text{sgn}([CB_c^-]_{ij} - D_{ij}) = 0$ and $\text{sgn}([CB_c^+]_{ij} - D_{ij}) = 1$. Moreover, $\text{sgn}([CB_c^-]_{ij} - [CB_c^+]_{ij}) = 1$. Therefore, $\text{sgn}([CB_c^+]_{ij} - [CB_c^-]_{ij}) = \text{sgn}([CB_c^-]_{ij} - D_{ij}) + \text{sgn}([CB_c^+]_{ij} - D_{ij})$.

2) If $[CB_c^+]_{ij} < [CB_c^-]_{ij}$, we have $D_{ij} = [CB_c^+]_{ij}$, $\text{sgn}([CB_c^-]_{ij} - D_{ij}) = 1$ and $\text{sgn}([CB_c^+]_{ij} - D_{ij}) = 0$. Moreover, $\text{sgn}([CB_c^-]_{ij} - [CB_c^+]_{ij}) = 1$. Therefore, $\text{sgn}([CB_c^+]_{ij} - [CB_c^-]_{ij}) = \text{sgn}([CB_c^-]_{ij} - D_{ij}) + \text{sgn}([CB_c^+]_{ij} - D_{ij})$.

3) If $[CB_c^+]_{ij} = [CB_c^-]_{ij}$, we have $D_{ij} = [CB_c^-]_{ij} = [CB_c^+]_{ij}$, $\text{sgn}([CB_c^-]_{ij} - D_{ij}) = 0$ and $\text{sgn}([CB_c^+]_{ij} - D_{ij}) = 0$. Moreover, $\text{sgn}([CB_c^-]_{ij} - [CB_c^+]_{ij}) = 0$. Therefore, $\text{sgn}([CB_c^+]_{ij} - [CB_c^-]_{ij}) = \text{sgn}([CB_c^-]_{ij} - D_{ij}) + \text{sgn}([CB_c^+]_{ij} - D_{ij})$.

As a result, the problem in (5.11) can be transformed to the following problem in (5.12) as follows.

$$
\begin{aligned}
&\arg\min_{C,D} \sum_{i=1}^{2d} \sum_{j=1}^{m} \text{sgn}(\left[CB_c^+\right]_{ij} - \left[CB_c^-\right]_{ij}),\\
=&\arg\min_{C,D} \sum_{i=1}^{2d} \sum_{j=1}^{m} \text{sgn}(\left[C\left(B_c^+ - B_c^-\right)\right]_{ij})\\
=&\arg\min \sum_{i=1}^{2d} \sum_{j=1}^{m} \text{sgn}([CB_c]_{ij}),
\end{aligned}
\tag{5.12}
$$

such that matrices C and D satisfy conditions (C1), (C2), and (C3).

In this thesis, we consider the Petri nets that have *state machine* structure (i.e., every transition in the Petri net model has only one input place and one output place). This is the common case in the transportation systems modeled by Petri nets where the transition is used to model the uni-directional passage between two locations (modeled by the places). Therefore, the incident matrix B of the original Petri net has the property that, every column of B has only one entry "1" and one entry "-1". All the rest elements in this matrix are zeros.

The control specifications in the transportation systems usually require certain location (place) can only hold a limited number of vehicles (modeled by the tokens). Reflected into the matrix L in [39], this implies that each row of L has only one nonzero entry. Moreover, since we usually have at most one control specification for each location (place), each column of L can have at most one nonzero entry. If a nonzero entries in L is not 1, we can make it and the corresponding element in the vector b introduced in [39] divided by itself. We call the matrix L in such form a *nearly identity matrix*. With such B and L, we can prove the following proposition for the incident matrix B_c of the original Petri net controller.

Proposition 5.3.2 *If the matrix B is the incident matrix of the state machine N and the matrix L for the given control specifications is a* nearly identity matrix*, the every column of the incident matrix B_c for the controller of N must belong to one of the following three cases.*

1. *The column is a 0-vector;*
2. *The column has only one nonzero element (either 1 or -1);*
3. *The column has only two nonzero elements (1 and -1).*

Proof Suppose $L_i (i = 1, 2, \ldots, n_c)$ is the i-th row of the matrix L and the j-th $(j = 1, 2, \ldots, n)$ element of L_i, i.e., $L_{ij} = 1$. Suppose $B_l (l = 1, 2, \ldots, m)$ is the l-th column of the matrix B.

From [39] we know the matrix $B_c = -LB$. Hence, the entry in the i-th row and l-th column of B_c, i.e., $[B_c]_{i,l} = -L_i B_l = -B_{jl}$. As a result, $[B_c]_{i,1} = -B_{j1}, [B_c]_{i,2} = -B_{j2}, \ldots, [B_c]_{i,m} = -B_{jm}$. We obtain that $[B_c]_i = -B_j$ where $[B_c]_i$ is the i-th row of B_c. Since each column of L can have at most one nonzero entry, each row in B_c corresponds to a distinct row in B. We treat B_C as the reordering of some rows of B and then multiplied by -1. Since every column of B has only one entry "1" and one entry "-1", we can easily conclude that each column of B must belong to one of the above three cases. ■

If one column $[B_c]_h$ of the matrix B_c belongs to Case 1 of Proposition 5.3.2, then $\mathrm{sgn}([CB_c]_{ih}) = 0 (i = 1, 2, \ldots, 2d)$ whatever the columns of the matrix C is interchanged. If one column $[B_c]_k$ of B_c belongs to Case 2 of Proposition 5.3.2 and $[B_c]_{lk} \neq 0$, then $\mathrm{sgn}([CB_c]_{jk}) = \mathrm{sgn}(C_{jl}) = 1 (j = 1, 2, \ldots, 2d)$ according to Proposition 5.2.3 and whatever the columns of C is interchanged. As a result, when we try to optimize Equation 5.12, the columns of B_c that belongs to Case 1 and 2 of Proposition 5.3.2 are not considered.

Suppose in the j-th column of B_c, the k-th element $[B_c]_{kj} = 1$ and the l-th element $[B_c]_{lj} = -1$. So $[CB_c]_{ij} = C_{ik} - C_{il}$ and $\mathrm{sgn}([CB_c]_{ij}) = \mathrm{sgn}(C_{ik} - C_{il})$, where C_{ik} and C_{il} are the k-th and l-th elements in the i-th row of C. Define

$$\begin{aligned} dif\ (k,l) =& \mathrm{sgn}(C_{1k} - C_{2l}) + \mathrm{sgn}(C_{2k} - C_{2l}) + \cdots + \\ & + \mathrm{sgn}(C_{2dk} - C_{2dl}) \\ =& \mathrm{sgn}([CB_c]_{1j}) + \mathrm{sgn}([CB_c]_{2j}) + \cdots + \\ & + \mathrm{sgn}([CB_c]_{2dj}) \end{aligned} \tag{5.13}$$

as the *distance* between the k-th column and l-th column of matrix C.

$\sum_{i=1}^{2d} \sum_{j=1}^{m} \mathrm{sgn}([CB_c]_{ij})$ is just the sum of such distances. For all columns in B_c, the number of times that the k-th and the l-th (for all $k, l \in \{1, 2, \ldots, n_c\}$ and $k \neq l$) elements are nonzero are not necessarily equal. For the purpose of minimization, we always hope that, if the appearance frequency of (k, l) (i.e., the k-th and l-th elements of B_c's one column are nonzero) is high, the distance between the k-th column and l-th column of C should be less. Based on this idea, we develop an approximation algorithm to obtain matrix C. Once matrix C is determined, matrix D can be derived from Proposition 5.3.1.

5.3.2 Algorithm Development

Define C^0 as the original C matrix where C_i^0 ($i \in \{1, 2, \ldots, n_c\}$) is the i-th column of C^0. Define C^f as the final C matrix where C_j^f ($j \in \{1, 2, \ldots, n_c\}$) is the j-th column of C^f. We treat C_i^0's as *objects* and C_j^f's as *positions* to hold objects. Both C_i^0 and C_j^f have three indicators: *color*, position set P, and related columns C_r. For C_i^0 (or respectively C_j^f), color *white* indicates this column has not been explored yet; color *gray* indicates this column has been explored but its precise position (or which object to be put in) has not been decided yet; color *black* indicates this column's precise position (or which object to be put in) has been decided. When color is *white*, $P = \phi$; when color is *gray*, P equals the indices of two possible positions (or objects); when color is *black*, P is the index of the corresponding position (or object). C_r is only used when color is *gray*. If $C_i^0.P = \{l, r\}$, then C_i^0 and $C_i^0.C_r$ together capture positions about C_l^f and C_r^f. C_r is set to ϕ in other two cases.

For simplicity, define $L = \begin{pmatrix} 2 \\ n_c \end{pmatrix} = \frac{n_c(n_c-1)}{2}$. Define $dif\,(i,j)\,(= dif\,(j,i))$ as the *distance* between columns C_i^0 and C_j^0. $Dif\,[1 \ldots L]$ is the reverse-sorted array of $dif\,(i,j)$'s and is called the *distance sequence*. $Dif\,[1]$ represents the largest one among $dif\,(i,j)$'s. $Dif^*\,[1 \ldots L^*]$ copies $Dif\,[1 \ldots L]$ initially, but its length L^* is dynamic. The *current index* of $Dif^*\,[1 \ldots L^*]$ is Dif^*_{in}.

$k\,(i,j)$ is the number of columns whose i-th and j-th elements are nonzero in B_c. It also indicates the frequency of subtraction operations between columns C_i^f and C_j^f. $K\,[1 \ldots L]$ is the sorted array of $k\,(i,j)$'s and is called the *frequency sequence*. $K\,[1]$ represents the least one among $k\,(i,j)$'s. The *current index* of $K\,[1 \ldots L]$ is K_{in}.

Based on the above analysis and variable definitions, now we present the algorithm as follows (we call it Algorithm 2). The five color-changing processes in C^0 (or C^f) are summarized in Table 5.1. Double-Black is not considered since there is no color-changing happening in this case.

Observation 5.3.1 *The algorithm has two nested* for *loops. In each iteration of the outer loop, we pick up $K\,[K_{in}]$ and want to find an element in $Dif^*\,[1 \ldots L^*]$ to match*

Algorithm 2 Fault-tolerant controller optimization

INPUT

- A Petri net controller $\mathcal{C}$ with n_c places and m transitions
- The input incident matrix B_c^-, the output incident matrix B_c^+
- The prime p
- The maximum number d of place faults that may have occurred

Compute $B_c = B_c^+ - B_c^-$ and $|B_c|$
Construct the original "C matrix" C^0
Construct distance sequence $Dif\,[1 \ldots L]$
Copy $Dif\,[1 \ldots L]$ to obtain $Dif^*\,[1 \ldots L^*]$
Construct frequency sequence $K\,[1 \ldots L]$
Set all $color \leftarrow white$, $P \leftarrow \phi$ and $C_r \leftarrow \phi$
$K_{in} \leftarrow 1$, $Dif^*_{in} \leftarrow 1$ and $B\_count \leftarrow 0$
for $K_{in} \leftarrow 1$ to m **Do do**
  for $Dif^*_{in} \leftarrow 1$ to L^* **Do do**
    $flag \leftarrow 0$
    if Color combinations of $Dif^*\,[Dif^*_{in}]$ and $K\,[K_{in}]$ match **then**
      Case 1 Double-White
        Paint the 4 columns *gray*
        Set 2 columns from C^f as P's of 2 columns from C^0 and vice versa
        Set each of the 2 columns from C^0 as C_r of another one and set each of the 2 columns from C^f as C_r of another one
        $flag \leftarrow 1$
      Case 2 Double-Gray If P checking matches
        Call Gray-Operation
        $flag \leftarrow 1.$
        $B\_count \leftarrow B\_count + 4$
      Case 3 Double-Black If P checking matches
        $flag \leftarrow 1$

Case 4 White-Gray If P checking of *gray* columns matches
Call White-Operation
Call Gray-Operation
$flag \leftarrow 1$
$B\_count \leftarrow B\_count + 3$
Case 5 White-Black If P checking of *black* columns matches
Call White-Operation
$flag \leftarrow 1$
$B\_count \leftarrow B\_count + 1$
Case 6 Gray-Black If P checking matches
Call Gray-Operation
$flag \leftarrow 1$
$B\_count \leftarrow B\_count + 2$
if $flag$ **then**
Remove $Dif^* \, [Dif^*_{in}]$ from $Dif^* \, [1 \ldots L^*]$
$L^* \leftarrow L^* - 1$
Break
if $B\_count == n_c$ **then**
Break

White-Operation
Paint the *white* columns from C^0 and C^f *black*
Set *white* column from C^f as P of *white* column from C^0 and vice versa
Return

Gray-Operation
Paint the *gray* columns from C^0 and C^f and their C_r *black*
Update *gray*'s P's according to corresponding relation
Set C_r's of both the *gray* columns and their original C_r as ϕ
Return

Table 5.1 Summary of color-changing processes

Color Combination Type	Column Color Change	White	Gray	Black
Double-White	$2W \to 2G$	-2	+2	0
Double-Gray	$4G \to 4B$	0	-4	+4
White-Gray	$1W + 2G \to 3B$	-1	-2	+3
White-Black	$1W + 1B \to 2B$	-1	0	+1
Gray-Black	$2G + 1B \to 3B$	0	-2	+2

it. For each element in $Dif^*[1 \ldots L^*]$ *(the current* $Dif^*[Dif^*_{in}]$*), we examine whether their color combination matches or not. If the color combination matches, we go to the corresponding color combination case in Algorithm 1. In each of the six color combination cases, we do* P *checking first, where the* P *checking between two gray columns from* C^0 *and* C^f *respectively is to check whether the index of each of the two columns belongs to another column's* P*, while the* P *checking between two black columns is to check whether the index of each of the two columns equals to another column's* P*. Once* $Dif^*[Dif^*_{in}]$ *matches* $K[K_{in}]$ *(we set* $flag = 1$*), we follow some color-changing operations and change indicators of the involved columns accordingly. We then remove* $Dif^*[Dif^*_{in}]$ *from* $Dif^*[1 \ldots L^*]$*, reduce the length of* $Dif^*[1 \ldots L^*]$ *by one and break out of the inner loop.* B_count *records the number of black columns in* C^0 *(or* C^r*). We then check whether every column in* C^0 *(or* C^f*) has become black. If it is true, we break out of the outer loop and the algorithm terminates.*

Observation 5.3.2 *In fact, there are at most* m *(recall that* m *is the number of columns in* B_c*) non-zero elements in* $K[1 \ldots L]$*. We only need to execute the outer loop for* m *times. The inner loop needs to be executed at most* $L, L-1, \ldots, L-m+1$ *times after each iteration of the outer loop. Recall that* $L = \begin{pmatrix} 2 \\ n_c \end{pmatrix} = \frac{n_c(n_c-1)}{2}$*. As a result, the computational complexity of Algorithm 1 can be obtained as* $O(n_c^2 m)$*.*

5.3.3 Proof of Algorithm Correctness

The following theorem, together with the two propositions, proves the completeness of the algorithm, i.e., for each $K[K_{in}]$ we are always able to find a suitable element in $Dif^*[1 \ldots L^*]$ to match it.

Proposition 5.3.3 *The number of white, gray, and black columns respectively in C^0 and C^f is always the same. The number of six kinds of elements (Double-White, Double-Gray, Double-Black, White-Gray, White-Black and Gray-Black) respectively in $Dif[1 \ldots L]$ and $K[1 \ldots L]$ is always the same.*

Proof Initially, C^0 and C^f contain only white columns; $Dif[1 \ldots L]$ and $K[1 \ldots L]$ contain only Double-White elements. Proposition 5.3.3 holds trivially.

Due to the color-matching judge, the color-changing processes that happen in $Dif[1 \ldots L]$ and $K[1 \ldots L]$ are the same. The number of white, gray and black columns respectively in C^0 and C^f always keeps the same. The number of six kinds of elements in $Dif[1 \ldots L]$ and $K[1 \ldots L]$ is still the same, too. ■

Proposition 5.3.4 *$Dif^*[1 \ldots L^*]$ contains all the Double-White, White-Gray and White-Black elements in $Dif[1 \ldots L]$ and $K[K_{in} \ldots L]$ contains all the Double-White, White-Gray and White-Black elements in $K[1 \ldots L]$.*

Proof Initially, $L^* = L$ and $K_{in} = 1$, so $Dif^*[1 \ldots L^*] = Dif[1 \ldots L]$ and $K[K_{in} \ldots L]$ $= K[1 \ldots L]$. Proposition 5.3.4 holds trivially.

None of the five color-changing processes will produce new white columns. When $Dif^*[Dif^*_{in}]$ matches $K[K_{in}]$, all involved columns from C^0 and C^f will not be white any more. Then $Dif^*[Dif^*_{in}]$ and $K[K_{in}]$ are removed from $Dif^*[1 \ldots L^*]$ and $K[K_{in} \ldots L]$. So $Dif^*[1 \ldots L^*]$ will still contain all the Double-White, White-Gray and White-Black elements in $Dif[1 \ldots L]$ and $K[K_{in} \ldots L]$ will still contain all the Double-White, White-Gray and White-Black elements in $K[1 \ldots L]$. ■

Theorem 5.3.1 *$K[K_{in}]$ is always able to find a suitable element in $Dif^*[1 \ldots L^*]$ to match it.*

Proof We prove Theorem 5.3.1 in three cases.

Case 1: $K[K_{in}]$ is Double-White.

From Proposition 5.3.3 and Proposition 5.3.4 we know that, there must exists a Double-White element in $Dif^*[1 \ldots L^*]$. Then from the Double-White case of the optimal algorithm, this element is suitable for $K[K_{in}]$.

Case 2: $K[K_{in}]$ is composed of a white column $C^f_{i'}$ and a non-white column $C^f_{j'}$.

We first claim that there must exists an element in $Dif[1 \ldots L]$ that is composed of a white column from C^0 and a column belonging to $C^f_{j'}.P$. If our claim is not true, then all columns in C^0 is non-white. This contradicts Proposition 5.3.3 since there is white columns in C^f. We then claim that such element of $Dif[1 \ldots L]$ must stay in $Dif^*[1 \ldots L^*]$ according to Proposition 5.3.4. Then this element is suitable for $K[K_{in}]$.

Case 3: $K[K_{in}]$ is composed of two non-white columns from C^f.

If $K[K_{in}]$ is Double-Black, according to the corresponding relation of P, there is only one element in $Dif[1 \ldots L]$ that is suitable for it. This element can not have already been explored, or $K[K_{in}]$ can not appear in $K[K_{in} \ldots L]$. If $K[K_{in}]$ is not Double-Black, according to the corresponding relation of P, we can obtain the elements in $Dif[1 \ldots L]$ that are suitable for $K[K_{in}]$. We claim that these elements from $Dif[1 \ldots L]$ can not have already been explored. If one of them has already been explored, $K[K_{in}]$ will be painted as Double-Black. So all of these elements stay in $Dif^*[1 \ldots L^*]$ and are suitable for $K[K_{in}]$. ■

5.4 An Illustrative Example

In this section, we use an AGV system proposed in [25] as our illustrative example. This system has a bi-directional merge flow-path layout shown in Fig. 5.1 (with solid arcs and bolded places). The corresponding Petri net model is shown in Fig. 5.2. Tokens represent vehicles. The places p_7, p_8, p_9 and p_{10} represent the zones that cover the intersection of two or more paths. The initial marking of the Petri net is $[1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$.

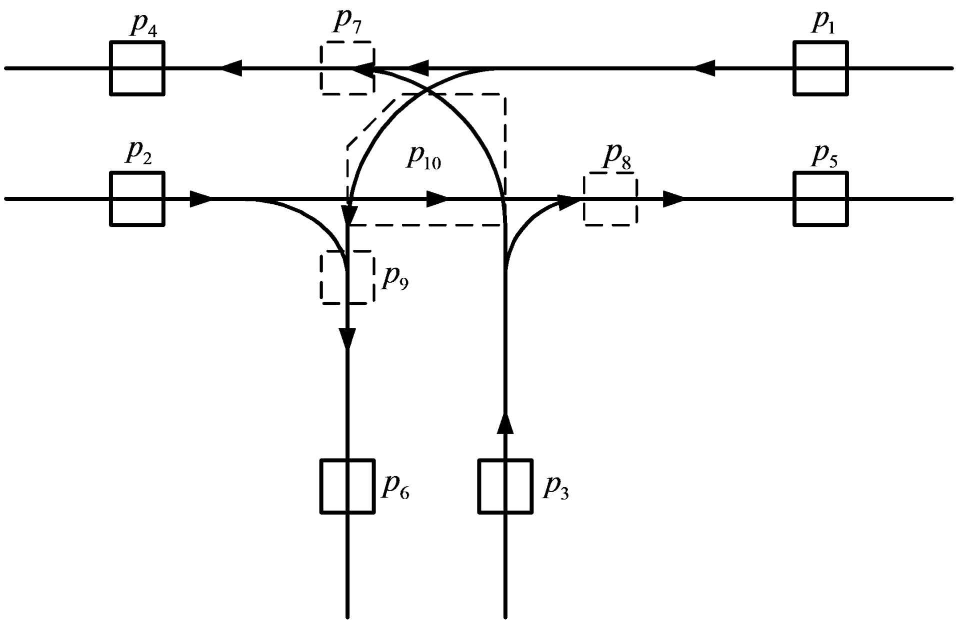

Fig. 5.1. The bi-direction merge flow-path layout

For the purpose of collision-free, we require that $M(p_7) \leq 1$, $M(p_8) \leq 1$, $M(p_9) \leq 1$, and $M(p_{10}) \leq 1$. As a result, the matrix L introduced in [39] is a *nearly identity matrix* and shown as follows.

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The vector b introduced in [39] is $b = [1\ 1\ 1\ 1]^T$. By enforcing the above constraints on the original system, a Petri net controller is derived based on the approach proposed in [39] and is shown in Fig. 5.3. In particular, places p_{c7}, p_{c8}, p_{c9} and p_{c10} are controller places and the dashed arcs are connections between these controller places and the transitions in the original system (to enforce the constraints mentioned above).

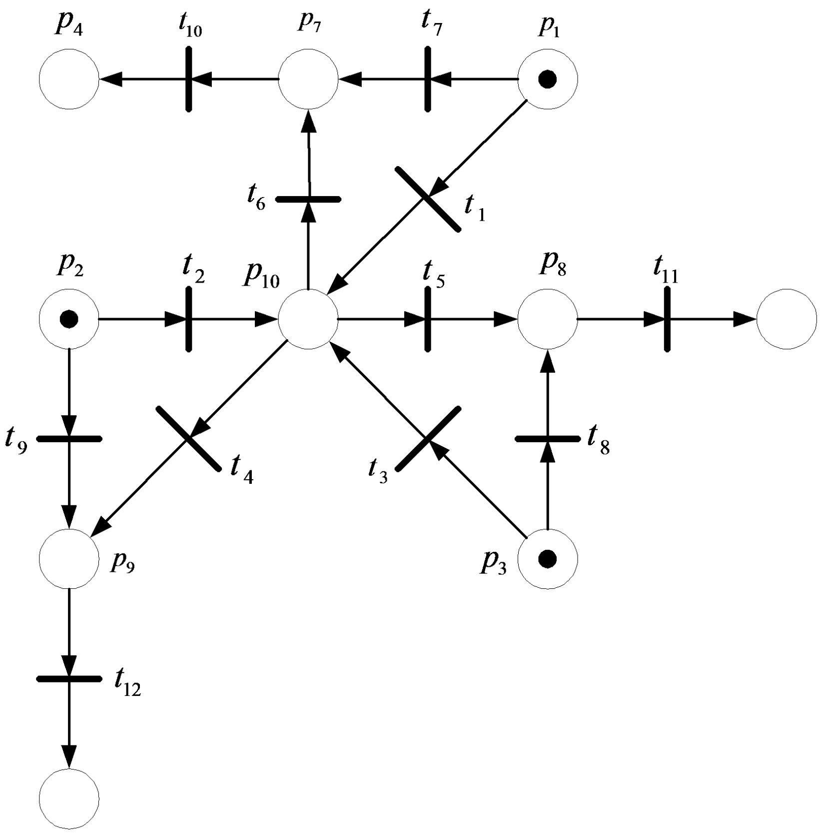

Fig. 5.2. The corresponding Petri net model for Fig. 5.1

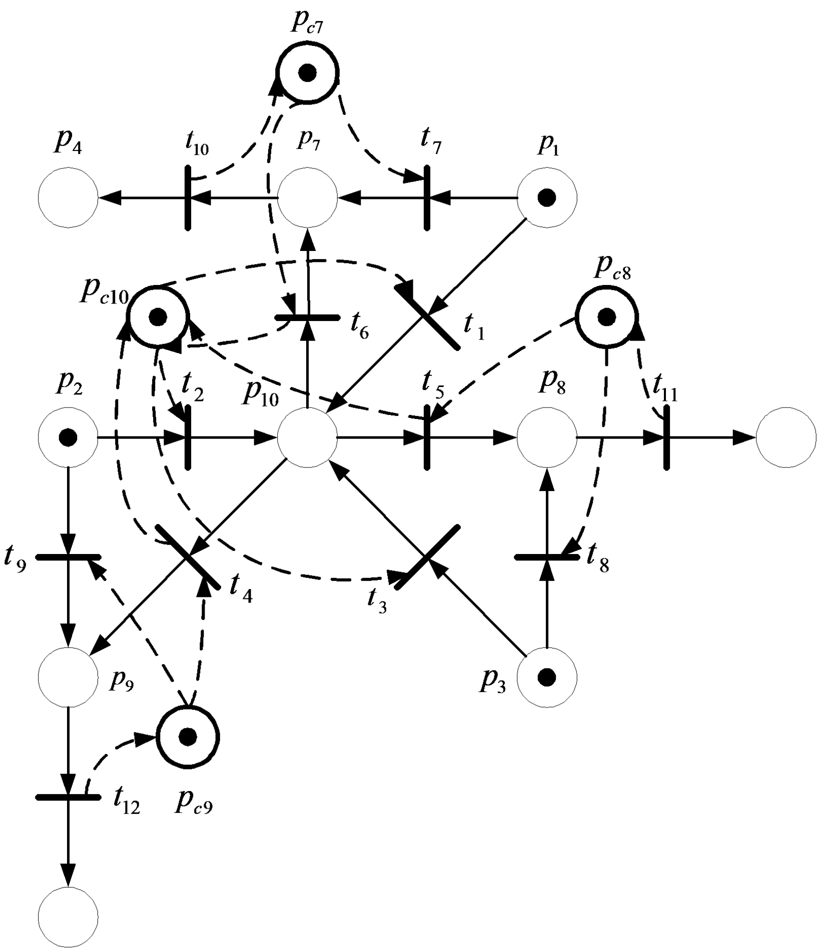

Fig. 5.3. The Petri net model with the controller

The initial marking of the controller is $M(P_{c7}) = 1$, $M(P_{c8}) = 1$, $M(P_{c9}) = 1$ and $M(P_{c10}) = 1$.

It is not difficult to see that the Petri net in Fig. 5.2 has a state machine structure. The incident matrix B_c of the Petri net controller is shown as follows.

$$B_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that there are four controller places, i.e., $n_c = 4$ and there are twelve transitions, i.e., $m = 12$. In this example, suppose we would like to detect and identify two place faults (i.e., we have $d = 2$), we pick up the prime p such that $p = 7 > n_c$. We randomly choose the original C matrix C^0 as follows ($\alpha_1 = 4$, $\alpha_2 = 2$, $\alpha_3 = 3$ and $\alpha_4 = 1$). Then we now get all necessary inputs for Algorithm 2.

$$C^0 = \begin{bmatrix} 4 & 2 & 3 & 1 \\ 2 & 4 & 2 & 1 \\ 1 & 1 & 6 & 1 \\ 4 & 2 & 4 & 1 \end{bmatrix},$$

$$dif\,(1,2) = 3,\ dif\,(1,3) = 2,\ dif\,(1,4) = 3,$$
$$dif\,(2,3) = 4,\ dif\,(2,4) = 3,\ dif\,(3,4) = 4,$$

$$Dif\,[1\ldots 6] = \begin{bmatrix} (2,3) & (3,4) & (1,2) & (1,4) & (2,4) & (1,3) \end{bmatrix},$$

$$k\,(1,4) = 1, k\,(2,4) = 1, k\,(3,4) = 1,$$
$$k\,(1,2) = 0, k\,(1,3) = 0, k\,(2,3) = 0,$$

$$K\,[1\ldots 6] = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (1,4) & (2,4) & (3,4) \end{bmatrix}.$$

Notice that we have omitted the columns in B_c that belongs to Case 1 and 2 of Proposition 5.3.2.

After the execution of Algorithm 2, we obtain that

$$C^f = \begin{bmatrix} 3 & 2 & 1 & 4 \\ 2 & 4 & 1 & 2 \\ 6 & 1 & 1 & 1 \\ 4 & 2 & 1 & 4 \end{bmatrix}.$$

With C^f, we obtain the results for (5.12) as

$$\sum_{i=1}^{2d}\sum_{j=1}^{m} \mathrm{sgn}([CB_c]_{ij}) = 44.$$

With C^0, we obtain the results for (5.12) as

$$\sum_{i=1}^{2d}\sum_{j=1}^{m} \mathrm{sgn}([CB_c]_{ij}) = 46.$$

It is clear that $44 < 46$ and after running the algorithm, we are able to obtain a fault-tolerant Petri net controller with a smaller number of arc weights when summed up, which illustrate the effectiveness of the proposed approach.

5.5 Summary

In this chapter, we proposed an approach for the design of fault-tolerant Petri net controllers for large-scale dynamic systems. In particular, we considered multiple faults detection and identification and developed an approximation algorithm to design such fault-tolerant controller to minimize the number of arcs in the redundant controller. An example of the fault-tolerant controller design for an AGV system was also provided to illustrate our approach.

6. SUMMARY

With the development of intelligent control strategies which simulate the human decision-making process and the improvement of the computation capability of micro controllers in recent years, DES have received considerable attention from both academy and industry. Among various DES models, Petri net is a hot research topic and has many practical applications. The graphical representation of Petri nets lends practitioners much convenience in modeling, analyzing, and controlling practical systems. The mathematical meaning under the graphical representation allow researchers to use Petri nets as the platform to study plenty of theories. Thus the research of Petri nets is quite important both in academy and industry.

6.1 Conclusions

In this thesis, we focused on three vital problems of Petri nets, namely, traffic system modeling, transition firing sequence reconstruction, and fault-tolerant controller optimization. For each problem, we used one chapter to discuss it. We conclude our work on these three problems separately as follows.

6.1.1 Signalized Intersection Modeling Based on Timed Petri Nets

In Chapter 3, a two-layer timed Petri net model was proposed for the signalized intersection in the microscopic sense. The first layer was the representation of the intersection and the second layer was the representation of the traffic light system. This model satisfied the modeling characteristics and requirements of the signalized intersection as we stated before. We listed the definitions of places and transitions in the above two Petri net representations. Based on these definitions, we described the cooperation process between the two Petri net representations to simulate and regulate the vehicle flow across the signalized intersection. The improvements of such model in describing all three kinds of turning behaviors and avoiding deadlocks, compared to the previous models, were also discussed.

6.1.2 Event Sequence Reconstruction of Sensor Networks Modeled by Petri Nets

In Chapter 4, we proposed a methodology for reconstructing possible transition firing sequences in a given Petri net based on asynchronous observations of the set of sequences of token changes in its places. The observation of each marking change sequence was assumed to be captured by a local sensor. Moreover, there was no global timing so that each sensor only captured the order of local marking changes. The original Petri net was partitioned into several subnets. The transition firing sequence of each subnet can be reconstructed through some special local observers. Based on the local observations from each sensor and each local observers, we developed an algorithm that was able to reconstruct all transition firing sequences that were consistent with these observations and the structure of the Petri net. The proposed algorithm proceeded in depth-first search fashion and iteratively reconstructed possible transition firing sequences. The complexity of the algorithm was discussed, too. An illustrative example was given to show the improvement of our algorithm compared to the previous algorithm.

6.1.3 Optimization of Fault-Tolerant Controllers for Petri Net Models

In Chapter 5, we proposed an approach for the design of fault-tolerant Petri net controllers for large-scale dynamic systems. Such redundancy was obtained through adding additional places and arcs to the original controller. The necessary and sufficient conditions for such redundant controllers not to interfere the normal operation of the original controllers were also provided. We devised the fault-tolerant controllers with multiple faults detection and identification ability. We developed an approximation algorithm to systematically design such fault-tolerant controller to minimize the number of arcs in the redundant controller. An example of the fault-tolerant controller design for an AGV system was also provided to illustrate our approach.

6.2 Future Work

Although our research about Petri nets in this thesis covers the complete process to apply Petri nets to practical systems, i.e., modeling, monitoring, and optimization, many research topics in this thesis are just the beginnings of a series of research work. Some extensions of our research topics will make our Petri net methodologies more suitable to practical systems. Other extensions may pioneer the new application fields of Petri nets. We list the main future research directions following the work in this thesis as follows.

6.2.1 Modular Modeling and Optimization of Traffic Networks

One of our future focuses on traffic system modeling is to model the road section and combine the models of the signalized intersections and road sections to construct the generic model of the traffic network. We will verify our model of the traffic network through the simulation with some urban traffic data sets. Another research direction on traffic system modeling is to apply some control strategies to achieve different optimization purposes, such as the minimization of total vehicular delay and the minimization of the traveling time of priority vehicles.

6.2.2 Optimal Division Strategy and Structure Utilization for Transition Firing Sequence Reconstruction

The local observer corresponding to each subnet of the Petri net model reduces the complexity of transition firing sequence reconstruction algorithm to a great extent.

If we can find certain optimal partition strategies of Petri nets, the complexity of the algorithm can be reduced further. However, the searching work of the optimal partition strategy for general Petri nets is quite labor-consumed. Then trying to find some optimal partition strategies for certain special subclasses of Petri nets is a nice entry point since we can make the best of the structural characteristics belonging to such subclasses of Petri nets. The experience on subclasses of Petri nets is also likely to lend us some inspiration for the research on general Petri nets.

6.2.3 Extension and Optimization of Fault-Tolerant Controller

Future extensions of the optimization algorithm for the fault-tolerant controllers include the development of optimization algorithms based on other practically meaningful criterions. We notice that the fault-tolerant ability of Petri net controllers is enabled under the coding of Petri net states (markings). Then another important future direction is to study other coding approaches (e.g., low-density parity-check codes) to explore more efficient ways for multiple faults detection and identification. Based on different coding schemes, we can develop other new optimization algorithms again.

LIST OF REFERENCES

LIST OF REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, vol. 11. Kluwer academic publishers, 1999.

[2] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[3] A. Bobbio, M. Gribaudo, and A. Horváth, "Modelling a car safety controller in road tunnels using hybrid petri nets," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pp. 1436–1441, IEEE, 2006.

[4] F. Diana, A. Giua, and C. Seatzu, "Safeness-enforcing supervisory control for railway networks," in *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on*, vol. 1, pp. 99–104, IEEE, 2001.

[5] H. Takahashi and K. Kuroda, "Intelligent vehicle control considering driver's visual perception," in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pp. 252–257, IEEE, 1999.

[6] N. Wu and M. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 6, pp. 1193–1202, 2005.

[7] T. Nishi and Y. Tanaka, "Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, no. 5, pp. 1230–1243, 2012.

[8] M. P. Fanti, "A deadlock avoidance strategy for agv systems modelled by coloured petri nets," in *Discrete Event Systems, 2002. Proceedings. Sixth International Workshop on*, pp. 61–66, IEEE, 2002.

[9] Y. Luo and J. Tsai, "A graphical simulation system for modeling and analysis of sensor networks," in *Multimedia, Seventh IEEE International Symposium on*, IEEE, 2005.

[10] R. AdiMallikarjuna, A. Kumar, and D. Janakiram, "e-petri net model for programming integrated network of wireless sensor networks and grids," in *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, pp. 1038–1043, IEEE, 2007.

[11] V. Devarashetty, J. J. Tsai, L. Ma, and D. Zhang, "Modeling a secure sensor network system using an extended elementary object system," in *Cognitive Informatics, 2008. ICCI 2008. 7th IEEE International Conference on*, pp. 67–74, IEEE, 2008.

[12] C.-H. Kuo and J.-W. Siao, “Petri net based reconfigurable wireless sensor networks for intelligent monitoring systems,” in *Computational Science and Engineering, 2009. CSE’09. International Conference on*, vol. 2, pp. 897–902, IEEE, 2009.

[13] D. He, L. Cui, H. Huang, and M. Ma, “Design and verification of enhanced secure localization scheme in wireless sensor networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 7, pp. 1050–1058, 2009.

[14] A. Shareef and Y. Zhu, “Energy modeling of wireless sensor nodes based on petri nets,” in *Parallel Processing (ICPP), 2010 39th International Conference on*, pp. 101–110, IEEE, 2010.

[15] J.-S. Lee, “A petri net design of command filters for semiautonomous mobile sensor networks,” *Industrial Electronics, IEEE Transactions on*, vol. 55, no. 4, pp. 1835–1841, 2008.

[16] L. Li, C. N. Hadjicostis, and R. S. Sreenivas, “Designs of bisimilar petri net controllers with fault tolerance capabilities,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 38, no. 1, pp. 207–217, 2008.

[17] Y. Qu, L. Li, Y. Chen, and Y. Dai, “Fault-tolerant controller design using petri nets with minimum initial state specifications,” in *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pp. 189–194, IEEE, 2010.

[18] Y. Qu, L. Li, Y. Chen, and Y. Dai, “Optimal design of fault-tolerant petri net controllers,” in *American Control Conference (ACC), 2010*, pp. 2607–2612, IEEE, 2010.

[19] J. Yan and L. Li, “Fault-tolerant controller design for automated guided vehicle systems based on petri nets,” in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 1531–1536, IEEE, 2012.

[20] L. Li and C. N. Hadjicostis, “Reconstruction of transition firing sequences based on asynchronous observations of place token changes,” in *Decision and Control, 2007 46th IEEE Conference on*, pp. 1898–1903, IEEE, 2007.

[21] G. Frey, “Automatic implementation of petri net based control algorithms on plc,” in *American Control Conference, 2000. Proceedings of the 2000*, vol. 4, pp. 2819–2823, IEEE, 2000.

[22] J. Júlvez and R. Boel, “Modelling and controlling traffic behaviour with continuous petri nets,” in *Proc. 16th IFAC World Congress*, 2005.

[23] J. Júlvez and R. K. Boel, “A continuous petri net approach for model predictive control of traffic systems,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 4, pp. 686–697, 2010.

[24] A. Di Febbraro, D. Giglio, and N. Sacco, “Urban traffic control structure based on hybrid petri nets,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 224–237, 2004.

[25] S. Hsieh and Y.-J. Shih, “Automated guided vehicle systems and their petri-net properties,” *Journal of Intelligent Manufacturing*, vol. 3, no. 6, pp. 379–390, 1992.

[26] A. Di Febbraro, D. Giglio, and N. Sacco, "Modular representation of urban traffic systems based on hybrid petri nets," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 866–871, IEEE, 2001.

[27] A. Di Febbraro and D. Giglio, "On representing signalized urban areas by means of deterministic-timed petri nets," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pp. 372–377, IEEE, 2004.

[28] A. Di Febbraro and D. Giglio, "On adopting a petri net-based switching modelling system to represent and control urban areas," in *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pp. 185–190, IEEE, 2005.

[29] A. Di Febbraro and D. Giglio, "Traffic-responsive signalling control through a modular/switching model represented via dtpn," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pp. 1430–1435, IEEE, 2006.

[30] A. Di Febbraro and D. Giglio, "Urban traffic control in modular/switching deterministic-timed petri nets," in *Control in Transportation Systems*, vol. 11, pp. 153–158, 2006.

[31] A. Di Febbraro, N. Sacco, and D. Giglio, "On using petri nets for representing and controlling signalized urban areas: New model and results," in *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pp. 1–8, IEEE, 2009.

[32] J. Yan, L. Li, and D. S. Kim, "Reconstruction of event sequences based on asynchronous observations in sensor networks," in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, ACM, 2013.

[33] S. Lin and D. J. Costello, "Error correcting coding: Fundamentals and applications," 1983.

[34] J. Yan and L. Li, "Microscopic modeling of a signalized intersection using timed petri nets." submitted, 2013.

[35] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Englewood Cliffs, NJ, 1995.

[36] Y. Wu and C. N. Hadjicostis, "Algebraic approaches for fault identification in discrete-event systems," *Automatic Control, IEEE Transactions on*, vol. 50, no. 12, pp. 2048–2055, 2005.

[37] Y. Wu and C. N. Hadjicostis, "Non-concurrent fault identification in discrete event systems using encoded petri net states," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 4, pp. 4018–4023, IEEE, 2002.

[38] U. Dudley, *Elementary number theory*. W. H. Freeman and Company, 1969.

[39] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, 1996.