STATE ESTIMATION AND SENSOR SELECTION
IN DISCRETE EVENT SYSTEMS MODELED BY PETRI NETS

BY

YU RU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Professor Tamer Ba¸sar, Chair
Adjunct Associate Professor Christoforos Hadjicostis
Associate Professor Daniel Liberzon
Associate Professor Ramavarapu Sreenivas
ABSTRACT

A discrete event system (DES) is a dynamic system that evolves in accordance with the abrupt occurrence, at possibly unknown and irregular intervals, of physical events. Such systems arise in a variety of contexts, such as energy distribution networks, computer and communication networks, automated manufacturing systems, air traffic control systems, highly integrated command, control, communication, and information (C3I) systems, advanced monitoring and control systems in automobiles or large buildings, intelligent transportation systems, and distributed software systems. Petri net models are widely used for modeling such systems, and consist of two key components: places (which typically model buffers that store system resources) and transitions (which typically model activities that move and process resources across places in the system). Sensors in Petri nets come in two major types: place sensors (i.e., sensors that indicate the number of resources in a particular place, e.g., vision sensors) and transition sensors (i.e., sensors that can detect whether a transition in a given subset of transitions has occurred, e.g., motion sensors).

In this dissertation, we focus on two sensor related problems in discrete event systems modeled by Petri nets:

(i) State estimation. When only transition sensors are available, sensor information can be very limited because there can be uncertainty due to unobservable events or events that generate the same sensor information. As a result, multiple states could be possible given sensing information, and we show in this dissertation that the number of possible states can grow at most polynomially (but not exponentially) as a function of the length of the observation sequence. These polynomial bounds can guide the design of systems, especially when trying to configure the sensors in order to reduce the uncertainty introduced in the state estimation stage. The polynomial bounds can also be used in
analyzing algorithms in the context of state estimation, fault diagnosis, supervisory control, and even reachability checking.

(ii) Sensor selection. If there are only transition sensors with uncertainty, the system state is usually not unique. If we have the freedom to configure sensors (e.g., when we design the system), we might want to add a minimal number of sensors to ensure that the current system state can be uniquely reconstructed based on the system model and the initial state. The design consideration is motivated by supervisory control applications, interface design for safety critical systems, and certain fault detection and correction settings. In its most general form, this type of sensor selection problem can involve both place sensors and transition sensors. We study how to choose a minimum number of place sensors and transition sensors (or a set of place sensors and transition sensors of minimal cost) while ensuring that the system state can be determined uniquely given sensing information and knowledge of the system model; this property is called structural observability. We show that the general sensor selection problem is computationally hard. If we are given a fixed set of transition sensors and are interested in selecting place sensors from a given set to achieve structural observability, the problem can be solved optimally by linear integer programming solvers, or suboptimally by heuristic methods we propose. On the other hand, if we have a fixed set of place sensors and then select transition sensors, the problem is solvable with complexity that is polynomial in the number of places and transitions. Among other potential applications, the heuristic methods we propose have implications for sensor selection to achieve immediate diagnosis of faults, reduct calculation in rough set theory, and approximating solutions for other NP-complete problems.
To my parents, for their love and support
ACKNOWLEDGMENTS

My first, and most earnest, acknowledgment goes to my Ph.D. adviser Professor Christoforos N. Hadjicostis. Chris has been very supportive over the last five years during my stay at the University of Illinois at Urbana-Champaign (UIUC), and has done his best in ensuring my academic, financial, and moral well being. In every sense, none of this work would have been possible without him. Many thanks also go to the other committee members, namely Professor Tamer Başar, Professor Daniel Liberzon, and Professor Ramavarapu Sreenivas for very helpful discussions.

Too many people to list individually have assisted me in different ways during my studies at UIUC. I express to all of them my sincere gratitude. In particular, I would like to thank our group members Eleftheria Athanasopoulou, Tung Le, Lingxi Li, Haitao Mo, Anooshiravan Saboori, Shreyas Sundaram, George Takos, and Rouzbeh Touri, and labmates Sourabh Bhattacharya, Michael Bloem, Praveen Bommannavar, Kien Nguyen, Hui Sun, Zhen Yin, and Quanyan Zhu, for very fruitful discussions. I would like to also acknowledge Professor Bruce Hajek, Professor R. Srikant, and Professor Sean Meyn for very helpful advice on different aspects of teaching. I would like to thank all of the instructors and fellow classmates in the courses I have participated in, either as a student or as a teaching assistant, ranging from Electrical Engineering and Industrial Engineering to Computer Science and English as a Second Language. My special thanks go to Ms. Amanda Huensch who has been consistently supportive of my English learning and teaching development. I would also like to thank many friends I met in different extracurricular activities such as OCEF (Oversea China Education Foundation) fund raising events, playing badminton and tennis, and traveling, among others.

I owe a huge debt of gratitude to Dr. Yueping Zhang, Dr. Guofei Jiang, and many other colleagues and friends at NEC Lab (Princeton, NJ), Prof.
Alessandro Giua, Dr. Maria Paola Cabasino, and many other researchers for very helpful discussions during conferences and visits. I would also like to thank anonymous reviewers of my journal and conference submissions, for they have provided invaluable comments which have made this work better.

This research has received funding from the National Science Foundation, Boeing, the European Commission, the IEEE Control Systems Society, and the College of Engineering (and Department of Electrical and Computer Engineering) at UIUC.

My final acknowledgment goes to my wonderful parents and my sister’s family. They have always been there when I needed them, and never complained about how infrequently I visited them; they deserve far more credit than I can ever give to them.
# TABLE OF CONTENTS

LIST OF TABLES ................................................. ix
LIST OF FIGURES .............................................. x
LIST OF ABBREVIATIONS ..................................... xii
LIST OF NOTATIONS ......................................... xiii

CHAPTER 1 INTRODUCTION .................................... 1

CHAPTER 2 PRELIMINARIES ................................... 6
  2.1 Petri Nets .............................................. 6
  2.2 Partially Observed and Partially Controlled Petri Nets .... 9
    2.2.1 Labeled Petri Nets ............................... 11
    2.2.2 Partially Observed Petri Nets ................. 13
  2.3 Computational Complexity Concepts .................... 14

CHAPTER 3 POLYNOMIAL BOUNDS ON THE NUMBER OF
CONSISTENT MARKINGS .................................... 17
  3.1 Introduction .......................................... 17
  3.2 Problem Formulation ................................... 18
  3.3 Upper Bounds on $|C(\omega)|$ for $\lambda$-free Labeled Petri Nets .... 20
  3.4 Upper Bounds on $|C(\omega)|$ for Petri Nets with Unobservable
    Transitions ............................................. 27
  3.5 Implications for State Estimation .................... 32
    3.5.1 State Estimation: Petri Nets without Unobservable
      Transitions ......................................... 33
    3.5.2 State Estimation: Petri Nets with Unobservable
      Transitions ......................................... 38
  3.6 Implications for Fault Diagnosis .................... 39
    3.6.1 Introduction ..................................... 39
    3.6.2 Problem Formulation ............................. 41
    3.6.3 Online Monitor for Belief Calculation .......... 42
    3.6.4 Extensions ...................................... 51
  3.7 Implications for Reachability Checking ............... 51
LIST OF TABLES

5.1 Simulation results of top-down method, bottom-up method, and combined method. ................. 103
5.2 Simulation results of SCP and BIP based methods. ........ 104
5.3 Comparison of heuristic methods with BIP based method over 30 simulations. ..................... 106
5.4 Simulation results for OPSS with costs. ................. 107
LIST OF FIGURES

1.1 Model-based approach to system analysis and synthesis. 3
2.1 Simple Petri net. 8
2.2 Labeled Petri net and its unobservable subnet. 13
2.3 A partially observed Petri net $Q$ configured with sensors. 14
3.1 Petri net analyzed in Example 3.1. 19
3.2 Computation of consistent markings for the net in Fig. 3.1. 20
3.3 Illustration of dividing balls. 24
3.4 Simple Petri net that is structurally bounded but not deadlock structurally bounded. 29
3.5 Plot of three quantities of interest against the length of the observed sequence of labels. 1 (green curve): number of consistent markings; 2 (blue curve): number of firing vectors; 3 (pink curve): upper bound obtained from Proposition 3.2. 35
3.6 Plot of the maximal number of consistent markings over 100 randomly generated observation sequences together with the polynomial upper bound of Theorem 3.1. 1 (green curve): maximal number of consistent markings; 2 (red curve): upper bound obtained from Theorem 3.1. 37
3.7 Petri net analyzed in Example 3.3. 39
3.8 Illustration of beliefs. 44
3.9 Calculation of $C_{\text{ext}}(\omega_e) = \{(M_9, K_9), (M_{10}, K_{10})\}$ from $C_{\text{ext}}(\omega) = \{(M_1, K_1), (M_2, K_2), (M_3, K_3), (M_4, K_4)\}$. 46
3.10 A simple strictly monotone Petri net. 57
4.1 Partially observed Petri net $Q$ with given labeling function $L$. 70
4.2 Partially observed Petri net $Q$ with given place sensor configuration $V$. 71
4.3 Reduction of MVC to OPSS. 73
5.1 Petri net model of a distribution network and its unobservable subnet. 98
5.2 Plot of the number of consistent markings and the upper bound obtained in Corollary 3.1 for the Petri net in Fig. 5.1(a) with the new labeling function $L'$. 1 (green curve): number of consistent markings given a random observation sequence; 2 (pink curve): number of consistent markings given a fixed observation sequence; 3 (red curve): upper bound obtained from Corollary 3.1.

5.3 A flexible manufacturing cell and its Petri net model.

5.4 Plot of running time.

5.5 Plot of the number of sensors resulting by each of the five methods.
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGV</td>
<td>Automated guided vehicle</td>
</tr>
<tr>
<td>BIP</td>
<td>Binary integer programming</td>
</tr>
<tr>
<td>C3I</td>
<td>Command, control, communication, and information</td>
</tr>
<tr>
<td>CGSS</td>
<td>Constrained general sensor selection</td>
</tr>
<tr>
<td>COTSS</td>
<td>Constrained optimal transition sensor selection</td>
</tr>
<tr>
<td>DES</td>
<td>Discrete event system</td>
</tr>
<tr>
<td>MVC</td>
<td>Minimum vertex cover</td>
</tr>
<tr>
<td>OPSS</td>
<td>Optimal place sensor selection</td>
</tr>
<tr>
<td>OTSS</td>
<td>Optimal transition sensor selection</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable logic controller</td>
</tr>
<tr>
<td>SCP</td>
<td>Set cover problem</td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$</td>
<td>A</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>The Cartesian product of sets $A$ and $B$</td>
</tr>
<tr>
<td>$\binom{n}{r}$</td>
<td>The binomial coefficient “$n$ choose $r$”</td>
</tr>
<tr>
<td>$\mathcal{C}(\omega)$</td>
<td>The set of markings consistent with observation sequence $\omega$</td>
</tr>
<tr>
<td>$D$</td>
<td>The incidence matrix of Petri net $G$</td>
</tr>
<tr>
<td>$D(:,t)$</td>
<td>The column of $D$ that corresponds to transition $t$</td>
</tr>
<tr>
<td>$D(p,:)$</td>
<td>The row of $D$ that corresponds to place $p$</td>
</tr>
<tr>
<td>$D_V$</td>
<td>A submatrix of $D$ obtained by keeping the rows of $D$ that correspond to observable places with sensors in $V$</td>
</tr>
<tr>
<td>$D^e_V$</td>
<td>A submatrix of $D_V$ obtained by keeping the columns in $D_V$ that correspond to transitions associated with label $e$</td>
</tr>
<tr>
<td>$e^k$</td>
<td>$k$ instances of label $e$, i.e., $\underbrace{e \cdot \cdot \cdot e}_{k \text{ times}}$</td>
</tr>
<tr>
<td>$f(p)$</td>
<td>Scoring function for an observable place $p$</td>
</tr>
<tr>
<td>$F$</td>
<td>The set of arcs of Petri net $G$</td>
</tr>
<tr>
<td>$G$</td>
<td>$G = \langle N, M_0 \rangle$: a Petri net with structure $N$ and initial marking $M_0$</td>
</tr>
<tr>
<td>$(G, \Sigma, L)$</td>
<td>Labeled Petri net: a Petri net $G$ with alphabet $\Sigma$ and labeling function $L$</td>
</tr>
<tr>
<td>$H$</td>
<td>An undirected graph</td>
</tr>
<tr>
<td>iff</td>
<td>If and only if</td>
</tr>
<tr>
<td>$k =</td>
<td>\omega</td>
</tr>
<tr>
<td>$k_e(\omega)$</td>
<td>The number of times label $e$ appears in observation sequence $\omega$ of length $k$</td>
</tr>
</tbody>
</table>
The labeling function of a labeled Petri net

$L_{max}$ Labeling function satisfying $L_{max}(t) = t$ for any $t \in T_o$ and $L_{max}(t) = \lambda$ for any $t \in T_{uo}$

$l$ The maximum number of transitions that can be associated with a nondeterministic label

$l_e$ The minimum number of transitions that can be associated with a nondeterministic label

The number of transitions that are associated with label $e \in \Sigma \cup \{\lambda\}$

$\lambda$ The null (or empty) label

$m$ The number of transitions in Petri net $G$

$	ext{max}\{a, b\}$ The maximum of $a$ and $b$

$	ext{min}\{a, b\}$ The minimum of $a$ and $b$

$M$ Marking of Petri net $G$

$M(p)$ The number of tokens in place $p$ at marking $M$

$M_0$ Initial marking of Petri net $G$

$M[t]$ Transition $t$ is enabled at marking $M$

$M[t]M'$ Marking $M'$ is reached after the firing of transition $t$ at marking $M$

$M[S]$ Transition sequence $S$ is enabled at marking $M$

$M[S]M'$ Marking $M'$ is reached after the firing of sequence $S$ at marking $M$

$\emptyset$ The empty set

$N$ Structure of Petri net $G = \langle N, M_0 \rangle$

$N_\lambda$ Unobservable subnet of a labeled Petri net $(G, \Sigma, L)$

$N_{T_{uo}}$ Unobservable subnet of a partially observed and partially controlled Petri net $Q$

$N_{T_{uc}}$ Uncontrollable subnet of a partially observed and partially controlled Petri net $Q$

$\mathcal{N}$ The set of positive integers

$\mathcal{N}_0$ The set of nonnegative integers
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number of places in Petri net ( G )</td>
</tr>
<tr>
<td>( \omega )</td>
<td>The observation sequence</td>
</tr>
<tr>
<td>( \Omega_e(p) )</td>
<td>Partition of ( T_e ) generated by observable place ( p )</td>
</tr>
<tr>
<td>( \Omega(V) )</td>
<td>Partition of ( T_o ) (or ( T )) generated by place sensor configuration ( V )</td>
</tr>
<tr>
<td>( O(\cdot) )</td>
<td>The big ( O ) notation in computational complexity theory</td>
</tr>
<tr>
<td>( \otimes )</td>
<td>An associative abstract operation</td>
</tr>
<tr>
<td>( P )</td>
<td>The set of places of Petri net ( G )</td>
</tr>
<tr>
<td>( P_o )</td>
<td>Observable places: the set of places that can be configured with sensors</td>
</tr>
<tr>
<td>( P_{uo} )</td>
<td>Unobservable places: the set of places with no sensor available</td>
</tr>
<tr>
<td>( \prod i k_i )</td>
<td>The product of all ( k_i )'s</td>
</tr>
<tr>
<td>( \cdot p )</td>
<td>The set of input transitions of place ( p )</td>
</tr>
<tr>
<td>( p^* )</td>
<td>The set of output transitions of place ( p )</td>
</tr>
<tr>
<td>( Q )</td>
<td>Partially observed (and partially controlled) Petri net</td>
</tr>
<tr>
<td>( R^+_0 )</td>
<td>The set of nonnegative real numbers</td>
</tr>
<tr>
<td>( R(G, M_0) )</td>
<td>The reachability set of Petri net ( G = \langle N, M_0 \rangle )</td>
</tr>
<tr>
<td>( S )</td>
<td>Firing sequence</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>The set of labels (or alphabet)</td>
</tr>
<tr>
<td>( \Sigma^* )</td>
<td>The set of all possible strings generated from the alphabet ( \Sigma )</td>
</tr>
<tr>
<td>( \Sigma_i k_i )</td>
<td>The sum of all ( k_i )'s</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Firing vector</td>
</tr>
<tr>
<td>( \sigma(t) )</td>
<td>The number of firings of transition ( t ) in firing sequence ( S )</td>
</tr>
<tr>
<td>( T )</td>
<td>The set of transitions of Petri net ( G )</td>
</tr>
<tr>
<td>( T_e )</td>
<td>The set of transitions associated with the label ( e \in \Sigma \cup {\lambda} )</td>
</tr>
<tr>
<td>( T_o )</td>
<td>Observable transitions: the set of transitions that can be configured with sensors</td>
</tr>
<tr>
<td>( T_{uo} )</td>
<td>Unobservable transitions: the set of transitions with no sensor available</td>
</tr>
</tbody>
</table>
• \( t \) The set of input places of transition \( t \)

\( t^* \) The set of output places of transition \( t \)

\( U \) The control set

\( u \) The control value

\( UR(M) \) The unobservable reach from marking \( M \) in a labeled Petri net \( (G, \Sigma, L) \)

\( V \) Place sensor configuration

\( V_{max} \) An \( n_1 \)-dimensional vector \( 1_{n_1} \) with all entries being 1

\( ||V|| \) The number of sensors in \( V \)

\( W \) Weight function of Petri net \( G \)

\( x! \) The factorial of nonnegative integer \( x \)

\( [x] \) The ceiling function

\( \lfloor x \rfloor \) The floor function

\( y^T \) The transpose of a vector (or matrix) \( y \)

\( \mathcal{Z} \) The set of integers

\( := \) Defined to be
CHAPTER 1

INTRODUCTION

Over the past few decades, the rapid evolution of computing, communication, and sensor technologies has brought about the proliferation of “new” dynamic systems, mostly artificially constructed and often highly complex, such as energy distribution networks, computer and communication networks, automated manufacturing systems, air traffic control systems, highly integrated command, control, communication, and information (C3I) systems, advanced monitoring and control systems in automobiles or large buildings, intelligent transportation systems, distributed software systems, and so forth [1]. In such man-made systems, most activities are governed by operational rules, and their dynamics are therefore characterized by occurrences of discrete events, some of which can be observed by sensors (e.g., vision sensors, motion sensors, radar) and some of which are unobservable (e.g., no sensor exists for such events given current sensor availability). In addition, some of these events can be controlled by human operators (e.g., the turning of a switch, or the sending of a packet) and some of them cannot be controlled (e.g., events in chemical reactions which are mostly uncontrollable, or actuator failures).

The term discrete event systems (DESs) captures dynamic systems that evolve in accordance with the abrupt occurrence, at possibly unknown and irregular intervals, of physical events [2]. Besides the systems mentioned above, for many applications of interest (e.g., supervisory control and fault diagnosis), many other complex systems with both discrete and continuous dynamics\(^1\) can also be abstracted as discrete event systems at a higher level.

In the study of DESs, there are three levels of abstraction based on the features of events (for a more detailed discussion, refer to [1]):

- Untimed (or logical): at this level, one only cares about the relative ordering of event occurrences.

\(^1\)Note that such complex systems could also be modeled as switched systems or hybrid systems, and then be analyzed using corresponding techniques.
• Timed: at this level, one cares about both the relative ordering of event occurrences and the exact times at which these events occur.

• Stochastic: at this level, one cares not only about the exact times and relative ordering of event occurrences, but also about statistical information regarding successive occurrences of events.

The choice of the appropriate level of abstraction typically depends on the objective of system analysis. For example, if one is interested in how long it will take for the system to reach a certain state, then one needs to consider the timed level of abstraction; if one is interested in the average number of parts in an input conveyor, then one needs to consider the stochastic level of abstraction. In this dissertation, we study problems related to the following questions:

• What are possible system states given a sequence of observation?

• How can we configure sensors such that the system state can be reconstructed?

• Is a specific state reachable or not in a given DES?

Since these problems do not require information regarding the exact times at which events occur (not to mention stochastic information), we focus on the untimed level of abstraction of DESs. Two types of models are particularly suitable: automata [3] (or equivalently, finite state machines) and Petri nets [4,5]. Automata are intuitive and easy to analyze in the finite state case, but they lack structure and thus may lead to very large state spaces when modeling complex systems. Compared to automata, Petri nets have certain advantages, such as a higher language complexity, a compact, structural, and graphical description of the state space, and the ability to synthesize in a modular way [6].

Given a discrete event system (e.g., a robotic manufacturing cell consisting of an input conveyor, an output conveyor, a robot, and a milling machine, as shown in Fig. 1.1) configured with various kinds of sensors (e.g., motion sensors or vision sensors), we could perform system analysis and/or synthesis based on sensing information by feeding the information about the observed events into a Petri net model, and by analyzing the compatibility of sensing information against the known model (e.g., to estimate possible system
In this dissertation, we focus on two sensor related problems in discrete event systems modeled by Petri nets: (i) state estimation, and (ii) sensor selection. We first clarify the nature of sensors we consider. As we will see in Chapter 2, there are two key components in Petri nets: places and transitions (refer to Definition 2.1). For example, in the robotic manufacturing cell in Fig. 1.1, places can model the input conveyor, and they can hold workpieces (called tokens in Petri nets); transitions on the other hand can be used to model activities such as the robot moving a workpiece from the input conveyor to the machine input buffer. As a result, sensors in Petri nets can be classified into two major types: place sensors (i.e., sensors that can monitor the number of tokens in a particular place, e.g., vision sensors) and transition sensors (i.e., sensors that can detect whether a transition in a given subset of transitions has occurred, e.g., motion sensors).

In Chapter 3, we consider the state estimation problem, assuming the absence of place sensors. Uncertainty arises due to unobservable events or
events that generate the same sensor information; one motivational problem is fault diagnosis as usually there is limited information available from sensors and multiple states might be possible given a sequence of observations. In this dissertation, we study how the number of states changes when more observations become available, especially the problem of bounding the number of possible system states. We show that, under reasonable assumptions, the number of possible system states grows polynomially (but not exponentially) with the length of the observation sequence [7, 8]. The polynomial bounds we obtain can guide the design of systems, especially when trying to configure the sensors in order to reduce the uncertainty introduced in the state estimation stage. The polynomial bounds can also be used to analyze the computational complexity for least-cost firing sequence estimation [9], fault diagnosis [10, 11], supervisor synthesis to enforce an arbitrary finite set of forbidden states [12, 13], and reachability checking for certain classes of Petri nets [14].

If there are only transition sensors with uncertainty, the system state is usually not unique. When we have the freedom to configure sensors (e.g., when we design the system), we might want to add a minimal number of sensors (or a set of sensors of minimal cost) to ensure that the system state can be uniquely reconstructed based on the system model and the initial state. The resulting sensor selection problem is studied in Chapter 4 and is motivated by the necessity to know precisely the current system state, which is a typical requirement in supervisory control, interface design for safety critical systems, and certain fault detection and correction settings. In the sensor selection problem, both place and transition sensors may be available, and we study how to choose a minimum number of place sensors and/or transition sensors (or more generally, a set of place sensors and transition sensors of minimal cost) so that the system state can be determined uniquely given the sensor information and the system model; this property is called structural observability (refer to Definition 4.1). To gain a better understanding about the general sensor selection problem, we have studied two subproblems: the optimal place sensor selection (OPSS) problem which fixes transition sensors, and the optimal transition sensor selection (OTSS) problem which fixes place sensors [15–17]. We first establish that the OPSS problem is computationally hard by showing that the corresponding decision problem is \( \mathcal{NP} \)-complete via a polynomial reduction from the decision
version of the minimum vertex cover problem (refer to Problem 2.1) to this decision problem. We also show that the OPSS problem can be reduced with polynomial complexity to the binary integer programming problem (refer to Problem 2.2), which can be solved optimally using existing binary integer programming solvers (at least for small problem instances). As an alternative to the binary integer programming-based approach, we also propose four approximation algorithms to approach the optimal solution. Unlike the OPSS problem, the OTSS problem can be solved efficiently in time that is polynomial in the number of places and transitions. Sensor selection problems (with constraints on the way transitions might share sensors) are also considered [18]. The heuristic methods that we propose have potential applications in sensor selection to achieve immediate diagnosis of faults [19], reduct calculation in rough set theory [20], and also approximating solutions to other \( \mathcal{NP} \)-complete problems (e.g., the minimum vertex cover problem).

The dissertation is organized as follows. In Chapter 2, basic concepts and notation used throughout the dissertation are reviewed, and partially controlled and partially observed Petri nets are introduced as a generalization of labeled Petri nets and partially observed Petri nets. State estimation is discussed in detail in Chapter 3 and the sensor selection problem is formulated and addressed in Chapter 4. The methods introduced in Chapters 3 and 4 are applied to some practical problems in Chapter 5. Chapter 6 concludes the dissertation and points out interesting future directions.
In this chapter, we review basic concepts and notation about Petri nets [4, 5], introduce partially observed and partially controlled Petri nets, and provide some pertinent discussions on computational complexity and \( \mathcal{NP} \)-complete problems [21].

2.1 Petri Nets

In this section, we review basic definitions of Petri nets. For more details, refer to [4, 5].

**Definition 2.1** A Petri net structure is a 4-tuple \( N = (P, T, F, W) \), where

- \( P = \{p_1, p_2, \ldots, p_n\} \) is a finite set of \( n \) places;
- \( T = \{t_1, t_2, \ldots, t_m\} \) is a finite set of \( m \) transitions;
- \( F \subseteq (P \times T) \cup (T \times P) \) is a set of arcs;
- \( W : F \rightarrow \mathcal{N} \) is a weight function, where \( \mathcal{N} \) is the set of positive integers;
- \( P \cap T = \emptyset \) and \( P \cup T \neq \emptyset \).

The set of all input places of a transition \( t \in T \) is defined as \( \cdot t = \{p \in P \mid (p, t) \in F\} \), and the set of all output places of a transition \( t \in T \) is defined as \( t^* = \{p \in P \mid (t, p) \in F\} \). Similarly, the set of all input transitions of a place \( p \in P \) is defined as \( \cdot p = \{t \in T \mid (t, p) \in F\} \), and the set of all output transitions of a place \( p \in P \) is defined as \( p^* = \{t \in T \mid (p, t) \in F\} \). A transition is a source transition if \( \cdot t = \emptyset \).
The incidence matrix $D$ is the $n \times m$ matrix with $D(i,j) = -W(p_i, t_j) + W(t_j, p_i)$.

We use $D(:, t)$ to denote the column of $D$ that corresponds to transition $t$, and $D(p, :)$ to denote the row of $D$ that corresponds to place $p$.

**Definition 2.2** A solution to the equation $DX = 0_n$, where $X$ is a nonzero vector with nonnegative integer entries, is called a *transition invariant*.

A *marking* is a function $M : P \rightarrow \mathcal{N}_0$ that assigns to each place a nonnegative integer number of tokens, where $\mathcal{N}_0$ is the set of nonnegative integers. We denote by $M(p)$ the number of tokens in place $p$.

**Definition 2.3** A Petri net $G = \langle N, M_0 \rangle$ is a Petri net structure $N$ with an initial marking $M_0$.

**Remark 2.1** In this dissertation, a Petri net may refer to a Petri net structure or a Petri net; the meaning will be clear from the context.

A transition $t$ is said to be state-enabled (or enabled if control is not taken into account) at marking $M$ if each input place $p$ of $t$ is marked with at least $W(p, t)$ tokens, i.e., $\forall p \in \text{\textbullet} t, \: M(p) \geq W(p, t)$. We use $M[t]$ to denote the fact that $t$ is state-enabled at marking $M$. Note that source transitions are always state-enabled. A state-enabled transition $t$ may fire; its firing removes $W(p, t)$ tokens from each input place $p$ of $t$, and adds $W(t, p')$ tokens to each output place $p'$ of $t$, resulting in a marking $M'$; this is denoted by $M[t]M'$. In this dissertation, we assume that only one transition can fire at any instant.

A (possibly empty) $k$-length firing sequence from marking $M$ is a sequence of transitions $S = t_{s_1}t_{s_2}\cdots t_{s_k}, \: t_{s_i} \in T$, such that $M[t_{s_1}]M_1[t_{s_2}]M_2\cdots[t_{s_k}]M'$; this is denoted by $M[S]M'$, and we say $S$ is state-enabled at marking $M$ (denoted by $M[S]$). Given a firing sequence $S$, the firing vector $\sigma$ is the $m \times 1$ vector with its $i$th entry being the number of times transition $t_i$ appears in $S$. Using firing vector $\sigma$, the marking $M'$ can also be obtained using the following state equation:

$$M' = M + D\sigma.$$  \hspace{1cm} (2.1)

A marking $M'$ is reachable from $M$ if there exists a firing sequence $S$ such that $M[S]M'$. Given a Petri net, the set of all reachable markings from $M_0$

---

Note that if $W(p_i, t_j)$ or $W(t_j, p_i)$ is not defined for a specific place $p_i$ and transition $t_j$, it is taken to be 0.
is called the reachability set and is denoted by $R(G, M_0)$. If $\forall p \in P$ and $\forall M \in R(G, M_0)$, $M(p) \leq K$ for some positive integer $K$, then we say the Petri net is $K$-bounded (or simply, bounded). A Petri net is said to be safe if it is 1-bounded. A Petri net is said to be structurally bounded if it is bounded for any finite initial marking $M_0$. Structural boundedness is characterized by the following lemma [5].

**Lemma 2.1** A Petri net $G$ with incidence matrix $D$ is structurally bounded if and only if there exists an $n$-dimensional vector $y$ with strictly positive integer entries such that $y^T D \leq 0^T_m$, where $y^T$ denotes the transpose of $y$, $0^T_m$ denotes an $m$-dimensional column vector with all entries being 0, and the inequality is taken elementwise.

In this dissertation, we also need the concept of the acyclic Petri net.

**Definition 2.4** A Petri net is acyclic if it has no directed circuits.

**Example 2.1** Pictorially, places are represented by circles, transitions by bars, and tokens by black dots, as shown in Fig. 2.1. In this Petri net, $P = \{p_1, p_2\}$, $T = \{t_1, t_2\}$, $F = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (t_2, p_1)\}$, $W$ is 1 for all arcs in $F$, and $M_0 = (2 \ 0)^T$. Note that $t_1$ is state-enabled and if it fires, $M_1 = (1 \ 1)^T$ is reached. As the incidence matrix $D$ is

$$
\begin{pmatrix}
-1 & 1 \\
1 & -1
\end{pmatrix}
$$

and $(1 \ 1)D \leq 0^T_2$, the Petri net is structurally bounded. However, the Petri net is not acyclic as there is a circuit. If transition $t_2$ and its related arcs are removed, then the Petri net becomes acyclic.
2.2 Partially Observed and Partially Controlled Petri Nets

In this section we introduce partially observed and partially controlled Petri nets, which allow us to deal with partial observation from place sensors and/or transition sensors, and partially controlled inputs. The concept is generalized from partially observed Petri nets as defined in [11, 17], and partially observed and partially controlled Petri nets as defined in [13].

**Definition 2.5** A partially observed and partially controlled Petri net $Q$ is a 5-tuple $(N, P_o, T_o, T_c, M_0)$, where

- $N = (P, T, F, W)$ is a Petri net structure with $n$ places and $m$ transitions;
- $P_o \subseteq P$ is the set of observable places with cardinality $n_1$ satisfying $0 \leq n_1 \leq n$;
- $T_o \subseteq T$, is the set of observable transitions;
- $T_c \subseteq T$, is the set of controllable transitions;
- $M_0$ is the initial state.

In a partially observed and partially controlled Petri net, the set of places is partitioned as $P = P_o \cup P_u$: $P_o$ is the set of observable places, and $P_u$ is the set of unobservable places. Each observable place can have a sensor (e.g., a vision sensor) that indicates the number of tokens in that particular place; unobservable places cannot have such sensors. One can always rename places to ensure that the first $n_1$ places correspond to the observable places; thus, we take $P_o = \{p_1, p_2, ..., p_{n_1}\}$. A place sensor configuration $V$ is defined as a vector $(v_1 \ v_2 \ ... \ v_{n_1})^T$, where $v_i$ is a binary variable for $i = 1, 2, ..., n_1$: if no sensor is put on place $p_i$, then $v_i = 0$; otherwise, $v_i = 1$. In total, there could be $2^{n_1}$ place sensor configurations. We use $||V|| := \sum_{i=1}^{n_1} v_i \leq n_1$ to denote the total number of place sensors used on observable places.

In a partially observed and partially controlled Petri net, the set of transitions $T$ is partitioned in two distinct ways:

---

2For sets $A$, $B$, and $C$, $A = B \cup C$ means $A = B \cup C$ and $B \cap C = \emptyset$. 


\[ T = T_o \sqcup T_{uo} : T_o \text{ is the set of observable transitions, and } T_{uo} \text{ is the set of unobservable transitions.} \]

\[ T = T_c \sqcup T_{uc} : T_c \text{ is the set of controllable transitions, and } T_{uc} \text{ is the set of uncontrollable transitions.} \]

Observable transitions can have sensors (in the form of labels, e.g., motion sensors) that indicate whether a transition in a given subset of transitions has fired (several transitions can share the same label); in contrast, unobservable transitions cannot be directly observed given current sensor availability (no sensors exist for such transitions). The association between sensors and transitions is captured by a labeling function \( L: T \rightarrow \Sigma \cup \{\lambda\} \), which satisfies \( L(t) = \lambda \) for any \( t \in T_{uo} \). We define \( \Sigma \) so that, for each \( e \in \Sigma \), there exists \( t \in T_o \) satisfying \( L(t) = e \). Therefore, \( |\Sigma| \) is the total number of transition sensors in use and could be zero if no transition sensor is used. If \( L(t) = t \) for any \( t \in T_o \) and \( L(t) = \lambda \) for any \( t \in T_{uo} \), the mapping \( L \) is called a natural projection.\(^4\) If \( L(t) = e \in \Sigma \), the firing of transition \( t \) generates the event (or label) \( e \). Thus, if multiple transitions are assigned to the same label, their firings are not distinguishable solely by observing the label (this point will be illustrated in detail in Chapter 4.2).

Given a firing sequence \( S = t_{s_1}t_{s_2}\cdots t_{s_k} \), the corresponding observation sequence is
\[
\omega = L(S) := L(t_{s_1})L(t_{s_2})\cdots L(t_{s_k}),
\]
i.e., a string in \( \Sigma^* \).\(^5\) Note that the empty label \( \lambda \) does not appear in a nonempty observation sequence, and therefore the occurrence of unobservable transitions in an execution of a Petri net goes unrecorded. Due to the possible presence of unobservable transitions in the firing sequence \( S = t_{s_1}t_{s_2}\cdots t_{s_k} \), the observation sequence \( L(S) \) could have any length between 0 and \( k \).

**Definition 2.6** Given a partially observed and partially controlled Petri net \( Q \) with labeling function \( L \) and an observed sequence of labels \( \omega \), the set of consistent markings is \( \mathcal{C}(\omega) = \{ M \in \mathcal{N}_0^n \mid \exists S \in T^* : M_0[S]M \text{ and } L(S) = \omega \} \).

\(^3\)\(|A|\) denotes the cardinality of the set \( A \), i.e., the number of elements in \( A \).

\(^4\)The term “natural projection” is widely used in the context of automata, refer to [22] for details.

\(^5\)\( \Sigma^* \) is the set of all possible strings generated from the alphabet \( \Sigma \), including the empty string \( \lambda \).
The partitioning of $T$ into controllable and uncontrollable transitions indicates what transitions can be influenced by an external supervisor. Uncontrollable transitions are transitions that cannot be disabled by a supervisor. For example, state transitions in chemical reactions are usually uncontrollable; similarly, actuator failures can be modeled by uncontrollable transitions.

A controllable transition can be disabled by an external supervisor even if it is state-enabled. We take the set of possible control actions to be all subsets of $T_c$. More formally, we define the control set as

$$U = \{u \mid u \subseteq T_c\},$$

where $u$ is called a control value. A controllable transition $t$ is said to be control-disabled if $t \in u$. If a transition $t$ is state-enabled and is not control-disabled, it is enabled and can fire following the state equation (2.1).

To handle unobservable transitions, we need the concept of the unobservable subnet.

**Definition 2.7** Given a partially observed and partially controlled Petri net $Q$, we define the unobservable subnet as a net $N_{uo} = (P', T_{uo}, F', W')$, where $P' = \{p \in P \mid \exists t \in T_{uo} : p \in \cdot t \cup t\}$, $F'$ is the restriction of $F$ to $(P' \times T_{uo}) \cup (T_{uo} \times P')$, and $W'$ is the restriction of $W$ to $F'$.

To handle uncontrollable transitions, we need the concept of the uncontrollable subnet.

**Definition 2.8** Given a partially observed and partially controlled Petri net $Q$, we define the uncontrollable subnet as a net $N_{uc} = (P, T_{uc}, F', W')$, where $F'$ is the restriction of $F$ to $(P \times T_{uc}) \cup (T_{uc} \times P)$, and $W'$ is the restriction of $W$ to $F'$.

### 2.2.1 Labeled Petri Nets

In this subsection, we review basic concepts of labeled Petri nets. A labeled Petri net can be treated as a special partially observed and partially controlled Petri net $(N, P_o, T_o, T_c, M_0)$ in which $P_o = T_c = \emptyset$ and a labeling function $L$ is given. Because this class of Petri nets is very important...
when we consider the state estimation problem in Chapter 3, we discuss it separately following the approach in [4, 23].

Intuitively, labeled Petri nets can be thought of as Petri nets equipped with transition sensors. If a transition $t$ with a sensor fires, the associated sensor reports the information using a corresponding label; in such a case, transition $t$ is said to be observable. If there is no sensor associated with transition $t$, then $t$ is said to be unobservable. More formally, a labeled Petri net is defined as follows.

**Definition 2.9** A labeled Petri net is a 3-tuple $(G, \Sigma, L)$, where

- $G = \langle N, M_0 \rangle$ with $N = (P, T, F, W)$;
- $\Sigma$ is a given set of labels (also called alphabet);
- $L : T \rightarrow \Sigma \cup \{\lambda\}$ is the labeling function, which assigns to each transition $t \in T$ a label from a given alphabet $\Sigma$ or the empty label $\lambda$.

Note that the same label may be associated with more than one transition and unobservable transitions are labeled with the empty label $\lambda$. If transition $t$ is not associated with the empty label, we say that $t$ is nondeterministic when its label is shared with other transitions; otherwise, we say $t$ is deterministic. Similarly, a label $e \in \Sigma$ is nondeterministic if there exists more than one transition $t$ such that $L(t) = e$; otherwise, $e$ is deterministic. We use $T_e$ to denote the set of transitions associated with the label $e \in \Sigma \cup \{\lambda\}$. Note that $T_e \cap T_{e'} = \emptyset$ if $e \neq e'$. If all transitions are observable (namely, $T_\lambda = \emptyset$), we say that the labeling function is $\lambda$-free and the Petri net is a $\lambda$-free labeled Petri net.

If a transition with label $e \in \Sigma$ fires, then we observe the label $e$. Given a firing sequence $S = t_{s_1}t_{s_2}\cdots t_{s_k}$, we define the corresponding observation sequence as

$$\omega = L(S) := L(t_{s_1})L(t_{s_2})\cdots L(t_{s_k}),$$

in the same way as for partially observed and partially controlled Petri nets.

Now we introduce the definition of the set of consistent markings [23], which is a special case of Definition 2.6.

**Definition 2.10** Given a labeled Petri net $(G, \Sigma, L)$ with initial marking $M_0$ and observation sequence $\omega$, the set of consistent markings is $C(\omega) = \{M \in N_0^n \mid \exists S \in T^* : M_0[S]M$ and $L(S) = \omega\}$. 
To handle unobservable transitions, we define the unobservable subnet $N_\lambda$ of a labeled Petri net, which is a special case of Definition 2.7.

**Definition 2.11** Given a labeled Petri net $(G, \Sigma, L)$, we define the unobservable subnet as a net $N_\lambda = (P_\lambda, T_\lambda, F_\lambda, W_\lambda)$, where $P_\lambda = \{ p \in P \mid \exists t \in T_\lambda, p \in \cdot t \cup t^* \}$, $F_\lambda$ is the restriction of $F$ to $(P_\lambda \times T_\lambda) \cup (T_\lambda \times P_\lambda)$, and $W_\lambda$ is the restriction of $W$ to $F_\lambda$.

**Example 2.2** In the labeled Petri net shown on the left of Fig. 2.2, $\Sigma = \{a, b, c\}$, $L(t_1) = \lambda$, $L(t_2) = L(t_3) = b$, $L(t_4) = L(t_5) = a$ and $L(t_6) = c$. Here, label $c$ is deterministic, but labels $a$ and $b$ are nondeterministic. Given firing sequence $S = t_1t_2$, $M_1 = (1 0 0 1)^T$ is reached from the initial marking $M_0 = (0 2 0 0)^T$. The corresponding observation sequence is $L(S) = \lambda b = b$, and $C(b) = \{M_1, M_2\}$, where $M_2 = (0 1 0 1)^T$. The unobservable subnet of this labeled Petri net is shown on the right of Fig. 2.2.

2.2.2 Partially Observed Petri Nets

In this subsection we introduce partially observed Petri nets [17]. A partially observed Petri net is a special partially observed and partially controlled Petri net $(N, P_o, T_o, T_c, M_0)$ in which $T_c = \emptyset$ and $M_0$ is not specified, i.e., a 3-tuple $(N, P_o, T_o)$.

**Remark 2.2** We do not specify $M_0$ in the definition of partially observed Petri nets because in the sensor selection problems (refer to Chapter 4.2), we focus on structural properties that do not depend on the initial state $M_0$. □
Example 2.3 The net on the left of Fig. 2.3 is a partially observed Petri net. Places $p_1$, $p_2$ and $p_3$ are observable but place $p_4$ is unobservable (unobservable places are drawn as shadowed circles), i.e., $P_o = \{p_1, p_2, p_3\}$ and $P_{uo} = \{p_4\}$. All transitions except $t_5$ are observable (unobservable transitions are drawn as shadowed bars), i.e., $T_o = \{t_1, t_2, t_3, t_4\}$ and $T_{uo} = \{t_5\}$. The net on the right shows the Petri net configured with the place sensor configuration $V = (1 \ 1 \ 1)^T$ (note that places with sensors are drawn as red thick circles; in this case, we choose $p_1$, $p_2$ and $p_3$ to be equipped with sensors) and the labeling function $L$ satisfying $L(t_1) = a$, $L(t_2) = L(t_3) = b$, $L(t_4) = c$ and $L(t_5) = \lambda$.

Suppose $M_0 = (2 \ 0 \ 1 \ 0)^T$, and transition $t_3$ fires at $M_0$. Then, the system trajectory is $M_0[t_3]M_1$, where $M_1 = (0 \ 0 \ 2 \ 0)^T$. Given the place sensor configuration $V$ and the labeling function $L$, the available sensor information is

$$(2 \ 0 \ 1)^T \rightarrow b \rightarrow (0 \ 0 \ 2)^T,$$

where $\rightarrow$ denotes the temporal order of observations.■

2.3 Computational Complexity Concepts

In this section, we review basic concepts and notation from the field of computational complexity. For more details, refer to [21]. We first introduce the $O(\cdot)$ notation.

Definition 2.12 A function $f(n)$ is $O(g(n))$ if there exists constant $c > 0$ and constant integer $n_0 \geq 0$ such that $|f(n)| \leq c|g(n)|$ for all values of $n \geq n_0$.
\[ n \geq n_0. \]

The *time complexity function* for an algorithm expresses its time requirements by giving, for each possible input length, the largest amount of time needed by the algorithm to solve a problem instance of that size. A *polynomial time algorithm* is defined to be one whose time complexity function is \( O(f(n)) \) for some polynomial function \( f \), where \( n \) denotes the input length. In practical applications, the size \( n \) depends on the scheme encoding the input, and therefore usually some characteristic parameters of the input can be used. For example, if the input is a Petri net, then characteristic parameters could be the number of places, the number of transitions, the sum of tokens in the initial marking, and others.

In the field of computational complexity, a problem is called a *decision problem* if all problem instances are mapped to either “true” or “false.” A decision problem is said to be in the class \( \mathcal{NP} \) if it can be solved by a nondeterministic Turing machine (see [3]) in a number of time steps that is polynomial in the size of the problem. A decision problem is said to be \( \mathcal{NP} \)-hard if solving it in polynomial time would make it possible to solve all problems in the class \( \mathcal{NP} \) in polynomial time. If a problem is \( \mathcal{NP} \)-hard and is in \( \mathcal{NP} \), the problem is said to be \( \mathcal{NP} \)-complete.

In general, showing that a problem \( A \) is \( \mathcal{NP} \)-complete (based on polynomial reduction) requires four steps [21]:

1. Showing that problem \( A \) is in \( \mathcal{NP} \).
2. Selecting a known \( \mathcal{NP} \)-complete problem \( B \).
3. Constructing a transformation \( f \) (also called reduction) from \( B \) to \( A \).
4. Proving that \( f \) is a transformation of polynomial complexity.

These four steps will be used to establish the \( \mathcal{NP} \)-completeness of the optimal place sensor selection problem in Chapter 4. The problem \( B \) we select is the *minimum vertex cover* (MVC) problem, which is a well known \( \mathcal{NP} \)-complete problem. To introduce the problem, we first define an undirected graph.

**Definition 2.13** An *undirected graph* is a pair \( H = (Z, E) \) such that \( E \subseteq Z \times Z \). The elements of \( Z \) are the vertices of the undirected graph \( H \), and the elements of \( E \) are its edges.
Definition 2.14 Given an undirected graph $H = (Z, E)$, a subset $Z' \subseteq Z$ is a vertex cover for $H$ if for each edge $(u, v) \in E$, where $u, v \in Z$, at least one of $u$ and $v$ belongs to $Z'$.

Problem 2.1 (Minimum Vertex Cover: Decision Version) Given an undirected graph $H = (Z, E)$ and a positive integer $l \leq |Z|$, is there a vertex cover $Z'$ such that $|Z'| \leq l$?

The other two $\mathcal{NP}$-complete problems that play an important role are the binary integer programming problem [21] and the set cover problem [24].

Problem 2.2 (Binary Integer Programming (BIP)) Given (i) a $q \times s$ integer matrix $A \in \mathbb{Z}^{q \times s}$ where $\mathbb{Z}$ is the set of integers, (ii) a $q$-dimensional integer vector $b \in \mathbb{Z}^q$, and (iii) an $s$-dimensional nonnegative integer vector $c \in \mathbb{N}_0^s$, find a binary $s$-dimensional vector $x \in \{0,1\}^s$ to minimize $c^T x$ subject to $Ax \geq b$.

Problem 2.3 (Set Cover Problem (SCP)) Given a universe $\mathcal{U}$ of $q$ elements, a collection of subsets of $\mathcal{U}$, $\mathcal{S} = \{S_1, ..., S_k\}$, find a minimum number of subsets of $\mathcal{S}$ that cover all elements of $\mathcal{U}$. 

16
CHAPTER 3

POLYNOMIAL BOUNDS ON THE
NUMBER OF CONSISTENT MARKINGS

In this chapter, we focus on the state estimation problem. We first derive bounds on the number of markings that are consistent with an observed sequence of labels in a labeled Petri net, and then apply them to the analysis of state estimation, fault diagnosis, and reachability checking.

3.1 Introduction

In many discrete event systems, state information cannot be obtained directly due to limited sensor availability. As a result, in applications that require explicit state information (e.g., supervisory control [25–30], fault diagnosis [31–33]), the problem of state estimation becomes crucial. When the underlying DES model is a labeled Petri net, system states that can be reached after observing a sequence of labels are usually not unique due to the fact that transitions can be nondeterministic (i.e., different transitions can share the same label because they cause identical sensor output; refer to Chapter 2.2.1) and/or unobservable (i.e., transitions can be associated with the null label because they go undetected by system sensors; refer to Chapter 2.2.1) [23,34,35]. Establishing upper bounds on the number of system states is very important since the calculation and storage required for obtaining the set of system states is a critical issue in state observers that are designed for the purposes of supervisory control or fault diagnosis.

In this chapter, following a worst case analysis of the state estimation problem, we establish upper bounds on the number of system states that are consistent with the observation of a sequence of labels. In particular, we obtain upper bounds on the number of consistent states in labeled Petri nets that may possess nondeterministic and/or unobservable transitions. First, assuming that the given Petri net has no unobservable transitions, we show
that the number of markings consistent with the observation of a sequence of labels grows at most polynomially in the length of the observation sequence; this is true despite the fact that the number of firing sequences can increase exponentially with the length of the observation sequence. This result applies to general Petri nets without any restrictions on the structure of the Petri net or the nature of the labeling function. Then, we show that this bound can be extended to Petri nets with unobservable transitions under the assumption that the unobservable subnet is structurally bounded. Using these bounds, we show that the state estimation problem can be solved with complexity that is polynomial in the length of the observation sequence, without requiring any assumptions on nondeterministic transitions (such as the contact-free assumption which is key for [23]). The polynomial bounds can also guide the design of systems, especially when configuring the state transition sensors, to reduce the uncertainty introduced during the state estimation stage.

The chapter is organized as follows. In Section 3.2, we formulate problems on finding upper bounds on the number of consistent markings, and in Sections 3.3 and 3.4, we present upper bounds on the number of consistent markings. State estimation problems for Petri nets with nondeterministic transitions and unobservable transitions are discussed in Section 3.5 and are shown, under some mild assumptions on the structure of the unobservable subnet, to have computational complexity that is polynomial in the length of the observation sequence. The bounds are applied to analyzing fault diagnosis in Section 3.6 and reachability checking in Section 3.7. Section 3.8 summarizes the chapter.

3.2 Problem Formulation

The general problem we consider is the following: given a labeled Petri net with a known initial marking and a sequence of labels (generated by transition activity in the Petri net), we want to find an upper bound on the number of markings that are consistent with the observation sequence. Note that there will be at least one consistent marking since we assume that the observed sequence of labels is generated by activity in the Petri net.

To simplify the problem, we first consider $\lambda$-free labeled Petri nets (refer to Chapter 2.2.1 for the definition).
Problem 3.1 Consider a $\lambda$-free labeled Petri net $(G, \Sigma, L)$ with a known initial marking $M_0$. Given an observation sequence $\omega$ of length $k$ (which corresponds to an unknown underlying firing sequence $S = t_{s_1}t_{s_2}\cdots t_{s_k}$ such that $\omega = L(S)$), find a tight upper bound on the number of consistent markings $|C(\omega)|$.

Depending on the Petri net and the observation sequence, the exact number of consistent markings may increase, decrease or remain unchanged as we observe more and more labels. However, there are extreme cases in which the number of consistent markings always increases. One such case is discussed in the following example.

Example 3.1 Consider the Petri net in Fig. 3.1. There are three source transitions $t_1$, $t_2$ and $t_3$, all of which are associated with label $e$ and are always enabled. If we observe the sequence of labels $eee$, consistent markings can be computed by enumerating all firing sequences as illustrated in Fig. 3.2(a). The root of the tree is the initial marking $M_0 = (0\ 0\ 0)^T$. The number of leaves in the tree is equal to the number of firing sequences and increases exponentially with the length of the sequence. Nodes with the same color (or the same number) at the same level represent the same consistent marking. The enumeration of consistent markings shown in Fig. 3.2(b) is obtained after merging identical markings at each level. As illustrated in Fig. 3.2(b), the number of consistent markings is much less than the number of firing sequences.

For Petri nets with unobservable transitions, we consider the following restricted problem.

Problem 3.2 Consider a labeled Petri net $(G, \Sigma, L)$ with a known initial marking $M_0$ and a structurally bounded unobservable subnet$^1$ $N_X$. Given

---

$^1$For structurally bounded Petri nets, refer to Chapter 2.1; for unobservable subnets of labeled Petri nets, refer to Definition 2.11.
Figure 3.2: Computation of consistent markings for the net in Fig. 3.1.

an observed sequence of labels $\omega$ with length $k$ (which corresponds to an unknown underlying firing sequence $S$ that may include unobservable transitions and satisfies $\omega = L(S)$), find a tight upper bound on the number of consistent markings $|C(\omega)|$.

3.3 Upper Bounds on $|C(\omega)|$ for $\lambda$-free Labeled Petri Nets

In this section, we consider Problem 3.1 in Section 3.2. One way to bound the number of consistent markings is to first bound the number of tokens in every place. From the state equation (2.1), we have for every place $p$

$$M(p) = M_0(p) + D(p,:)\sigma .$$

Define $a_1$ to be the maximum entry of $M_0$, and $a_2$ to be the maximum entry of $D$; then, after observing a label sequence $\omega$ of length $k$,

$$M(p) \leq a_1 + a_2(1 \ 1 \ \cdots \ 1)\sigma = a_1 + a_2k,$$
where \((1 \ 1 \ \cdots \ 1)\sigma = k\) because there are \(k\) transition firings in any firing sequence \(S\) that can be mapped to \(\omega\).

**Lemma 3.1** Consider a \(\lambda\)-free labeled Petri net \((G, \Sigma, L)\) with a known initial marking \(M_0\). After observing a sequence of labels \(\omega\) of length \(k\), the number of consistent markings is upper bounded by

\[
(1 + a_1 + a_2 k)^n,
\]

where \(a_1\) is the maximum entry of \(M_0\), \(a_2\) is the maximum entry of \(D\), and \(n\) is the number of places in Petri net \(G\).

In general, the above straightforward bound is not tight (especially for Petri nets with a relatively large number of places) because

- many markings that satisfy \(M(p) \leq a_1 + a_2 k\) may not be reachable given the observed sequence of labels \(\omega\), and

- the bound depends on the maximal entry of the initial marking \(M_0\) while, intuitively, the number of consistent markings will only depend on \(\omega\) when \(M_0\) is large enough.

Another way to bound the number of consistent markings is to first bound the number of firing vectors because every consistent marking must be associated with at least one firing vector. Therefore, if we bound the number of firing vectors for the observed sequence of labels, we automatically have an upper bound on the number of consistent markings.

**Remark 3.1** Note that, given a firing vector \(\sigma\), there exists a unique marking \(M = M_0 + D\sigma\). However, given a marking \(M\), there may exist more than one corresponding firing vector. For example, if two transitions have the same input and output places and identical arc weights, then the firing of any one of them results in the same marking but different firing vectors. It is easy to verify that a sufficient condition for a given reachable marking to correspond to a unique firing vector is for the incidence matrix \(D\) to have full column rank.

Before considering the general case, we demonstrate our result for the net in Example 3.1. Clearly, \(|T_e| = 3\) where \(T_e = \{t_1, t_2, t_3\}\). If the observation
sequence is \( \omega = e^k := e e \cdots e \) \( k \) times (namely, \( k \) instances of label \( e \)), where \( k \geq 0 \), then

\[
\sigma(t_1) + \sigma(t_2) + \sigma(t_3) = k, \tag{3.1}
\]

with \( \sigma(t_i) \geq 0 \) for \( i = 1, 2, 3 \), denoting the number of firings of transition \( t_i \) in firing sequence \( S \) which can be mapped to firing vector \( \sigma \). Obviously, every firing vector \( \sigma \) must satisfy\(^2\) Eq. (3.1). Therefore, the number of solutions to Eq. (3.1) is an upper bound on the number of firing vectors and thus, also an upper bound on the number of markings that are consistent with the observation sequence \( e^k \).

Since \( \sigma(t_3) = k - (\sigma(t_1) + \sigma(t_2)) \), the number of solutions to Eq. (3.1) is equal to the number of all possible combinations of \( \sigma(t_1) \) and \( \sigma(t_2) \), which satisfy

\[
\sigma(t_1) + \sigma(t_2) \leq k.
\]

Equivalently, the above inequality can be expressed as the disjunction of the equalities

\[
\begin{align*}
\sigma(t_1) + \sigma(t_2) &= 0 \\
\sigma(t_1) + \sigma(t_2) &= 1 \\
&\vdots \\
\sigma(t_1) + \sigma(t_2) &= k - 1 \\
\sigma(t_1) + \sigma(t_2) &= k.
\end{align*}
\]

Therefore, the number of solutions is

\[
1 + 2 + \ldots + k + (k + 1) = \frac{(k + 1)(k + 2)}{2} = \binom{k + 2}{2},
\]

where \( \binom{n}{r} \) is the binomial coefficient “\( n \) choose \( r \)”. This is an upper bound on the number of markings that are consistent with the observation sequence \( e^k \).

\(^2\)In general, the converse (i.e., every solution of Eq. (3.1) must be a firing vector) does not hold because there may not exist an enabled transition sequence corresponding to some solution of Eq. (3.1); however, in this particular example, the converse does hold because all transitions are always enabled.
Similarly, if \( T_e = \{ t_1, t_2, t_3, t_4 \} \) and the observation sequence is still \( e^k \), an upper bound on the number of consistent markings is \( \binom{k+3}{3} \). Based on these observations, one can deduce the following proposition, which will be formally proved in Proposition 3.2 as a special case.

**Proposition 3.1** Consider a \( \lambda \)-free labeled Petri net \((G, \Sigma, L)\) with a known initial marking \( M_0 \) and all transitions labeled \( e \). If the observation sequence is \( e^k \), then the number of consistent markings is upper bounded by

\[
\binom{k + l_e - 1}{l_e - 1},
\]

where \( k \geq 0, l_e = |T_e| = |T| = m \geq 1 \), and \( m \) is the number of transitions in Petri net \( G \).

Now we turn to the general case. To simplify the representation, without loss of generality we assume that

- there are \( d \) labels \( e_1, e_2, \ldots, e_d \), where \( 1 \leq d \leq m \);
- \( l_{e_i} := |T_{e_i}| \geq 2 \) for \( i = 1, \ldots, j \) while \( l_{e_i} := |T_{e_i}| = 1 \) for \( i = j + 1, \ldots, d \), where \( 0 \leq j \leq d \);
- the observation is a sequence of labels \( \omega \) of length \( k \geq 0 \), in which \( e_i \) appears \( k_{e_i} \geq 0 \) times and \( k_{e_1} + \ldots + k_{e_d} = k \).

**Remark 3.2** Note that if \( j = 0 \), all labels are deterministic (and \( d = m \)); if \( j = d \), all labels are nondeterministic. Given the alphabet \( \Sigma \), in which \( d \) is the total number of labels and \( j \) is the number of nondeterministic labels, one can rename the nondeterministic labels to be \( e_1, e_2, \ldots, e_j \) and the deterministic labels to be \( e_{j+1}, \ldots, e_d \), to ensure that the above notation holds.

**Proposition 3.2** Consider a \( \lambda \)-free labeled Petri net \( G \) with a known initial marking \( M_0 \) as described above. If the observation is a sequence of labels \( \omega \) of length \( k \) in which label \( e_i \) appears \( k_{e_i} \) times and \( k_{e_1} + \ldots + k_{e_d} = k \), then the number of consistent markings is upper bounded by

\[
\prod_{i=1}^{j} \binom{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1},
\]

which is defined to be 1 if \( j = 0 \).
Proof: Given an observed sequence of labels $\omega$, a firing vector $\sigma$ corresponding to some marking $M \in C(\omega)$ satisfies the following set of equations:

$$
\begin{align*}
\sum_{t \in T_{e_1}} \sigma(t) &= k_{e_1} \\
& \vdots \\
\sum_{t \in T_{e_j}} \sigma(t) &= k_{e_j} \\
& \vdots \\
\sum_{t \in T_{e_j}} \sigma(t) &= k_{e_j} \\
\sigma(t_{e_j+1}) &= k_{e_j+1} \\
& \vdots \\
\sigma(t_{e_d}) &= k_{e_d} 
\end{align*}
$$

(3.2)

where $T_{e_i}$ is the set of transitions that are associated with label $e_i$ for $i = 1, 2, \ldots, j$ and $T_{e_i} = \{ t_{e_i} \}$ for $i = j + 1, \ldots, d$.

For each nondeterministic label $e_i$, $\sum_{t \in T_{e_i}} \sigma(t) = k_{e_i}$, where $\sigma(t) \geq 0$ for $t \in T_{e_i}$. Let $\beta(t) = \sigma(t) + 1$ for $t \in T_{e_i}$ so that $\beta(t) \geq 1$; we have

$$
\sum_{t \in T_{e_i}} \beta(t) = k_{e_i} + l_{e_i}.
$$

Now we can think of the problem of finding all possible values of $\beta(t)$ as the problem of dividing $k_{e_i} + l_{e_i}$ balls into $l_{e_i}$ groups while there should be at least one ball in each group. To arrange the $k_{e_i} + l_{e_i}$ balls into $l_{e_i}$ groups, we need to choose $l_{e_i} - 1$ separators among $k_{e_i} + l_{e_i} - 1$ separators (see Fig. 3.3, in which subindices in $k_{e_i}$ and $l_{e_i}$ are omitted for clarity). It is easy to verify that there are $\binom{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1}$ combinations; that is to say, there are $\binom{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1}$ possible solutions of $\sum_{t \in T_{e_i}} \sigma(t) = k_{e_i}$.

Therefore, the number of solutions to Eq. (3.2) (namely, the upper bound

Figure 3.3: Illustration of dividing balls.
on the number of consistent markings) is the product of all \(\left(\frac{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1}\right)\) for \(i = 1, 2, \ldots, j\), i.e., \(\prod_{i=1}^{j} \left(\frac{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1}\right)\), because the transition sets \(T_{e_1}, T_{e_2}, \ldots, T_{e_j}\) are pairwise disjoint. If \(j = 0\), then all labels are deterministic and there is only one consistent marking.

The upper bound in Proposition 3.2 requires the distribution of transitions with respect to labels (namely, the \(l_{e_i}\)’s) as well as specific information about the observation sequence (namely, the \(k_{e_i}\)’s). The following theorem relaxes this upper bound and expresses it in terms of structural parameters of the given labeled Petri net and the total length of the observation sequence (as opposed to the individual \(k_{e_i}\)’s).

**Theorem 3.1** Consider a \(\lambda\)-free labeled Petri net \((G, \Sigma, L)\) with a known initial marking \(M_0\). If the observation sequence \(\omega\) has length \(k\), then the number of consistent markings is upper bounded by

\[
\frac{(\frac{k}{j} + \frac{1}{2})^{j(l-1)}}{((l-1)!)^j},
\]

where \(j \geq 1\) is the number of nondeterministic labels, \(\bar{l} = \max\{l_{e_1}, l_{e_2}, \ldots, l_{e_j}\}\) and \(\underline{l} = \min\{l_{e_1}, l_{e_2}, \ldots, l_{e_j}\}\). If \(j = 0\), we define\(^3\) the value of Eq. (3.3) to be 1.

**Proof:** The bound holds for \(j = 0\) because in this case, all labels are deterministic and the number of consistent markings is 1. Now we prove the result for \(j \geq 1\).

Assume that \(k_{e_1} + \ldots + k_{e_j} = q\); obviously \(q \leq k\). Since

\[
\frac{(k_{e_i} + 1) \cdots (k_{e_i} + l_{e_i} - 1)}{(l_{e_i} - 1)!} \leq \frac{(k_{e_i} + 1) \cdots (k_{e_i} + \bar{l} - 1)}{(\bar{l} - 1)!}
\]

based on the definitions of \(\bar{l}\) and \(\underline{l}\) we get

\[
\prod_{i=1}^{j} \left(\frac{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1}\right) \leq \prod_{i=1}^{j} \frac{(k_{e_i} + 1) \cdots (k_{e_i} + \bar{l} - 1)}{(\bar{l} - 1)!}.
\]

\(^3\)Interestingly, if we treat Eq. (3.3) as a continuous function of \(j\), its limit using L’Hôpital’s Rule is also 1 as \(j\) goes to 0.
Note that for $1 \leq r \leq \ell - 1$, we have

$$(k_{e_1} + r) + \ldots + (k_{e_j} + r) = q + jr$$

and

$$(k_{e_1} + r) + \ldots + (k_{e_j} + r) \geq j \sqrt[j]{(k_{e_1} + r) \cdots (k_{e_j} + r)} \quad (3.5)$$

by the arithmetic-mean/geometric-mean inequality\(^4\) [36]. Therefore,

$$(k_{e_1} + r) \cdots (k_{e_j} + r) \leq \left( \frac{q + jr}{j} \right)^j$$

and

$$\prod_{i=1}^{j} \left( \frac{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1} \right) \leq \frac{(k_{e_1} + 1) \cdots (k_{e_j} + 1) \cdots (k_{e_1} + \ell - 1) \cdots (k_{e_j} + \ell - 1)}{((\ell - 1)!)^j}$$

$$\leq \frac{\prod_{r=1}^{\ell-1} (\frac{q + jr}{j})^j}{((\ell - 1)!)^j} = \left( \frac{\frac{q}{j} + \frac{q}{j} + 2 + \ldots + \frac{q}{j} - 1}{\ell - 1} \right)^j$$

$$\leq \left( \frac{\frac{q}{j} + \frac{q}{j} + 2 + \ldots + \frac{q}{j} - 1}{\ell - 1} \right)^j \quad (3.6)$$

$$= \frac{(q + \frac{q}{j} + 1)(q + \frac{q}{j} + 2) \cdots (q + \ell - 1)}{(\ell - 1)!^j} \leq \frac{(\ell + \frac{q}{j})^j(\ell - 1)}{(\ell - 1)!^j} \quad (3.7)$$

Equation (3.6) also follows from the arithmetic-mean/geometric-mean inequality, and Eq. (3.7) holds because $q \leq k$. 

Remark 3.3 In Eq. (3.3), parameters $j$, $\ell$ and $l$ depend solely on the labeling function. The upper bound on the number of consistent markings is polynomial in the length of the observation sequence $k$, i.e., it is $O(k^j(\ell - 1))$, despite the fact that the number of possible firing sequences can increase exponentially with the length of the observation sequence. 

\(^4\)For any $n$ nonnegative real numbers $x_1, x_2, \ldots, x_n$, the arithmetic-mean/geometric-mean inequality is $\frac{\sum_{i=1}^{n} x_i}{n} \geq \sqrt[n]{\prod_{i=1}^{n} x_i}$. 

26
3.4 Upper Bounds on $|\mathcal{C}(\omega)|$ for Petri Nets with Unobservable Transitions

In this section, we consider Problem 3.2 as defined in Section 3.2. One way to bound the number of consistent markings is to first bound the number of tokens in every place. For any $M \in \mathcal{C}(\omega)$, $M = M_0 + D\sigma \geq 0_n$, where $\sigma$ is a firing vector corresponding to $M$. Using the partition of unobservable transitions (namely, $T_\lambda$) and observable transitions (namely, $T_o = T \setminus T_\lambda$) in labeled Petri nets, the state equation of the places $P_\lambda$ (namely, the set of places in the unobservable subnet $N_\lambda$ as defined in Definition 2.11) can be written as

$$M^\lambda = M_0^\lambda + D_o^\lambda \sigma_o + D_{uo}^\lambda \sigma_{uo} \geq 0_{n_2}, \quad (3.8)$$

where $M^\lambda$ is the restriction of $M$ to the set of places $P_\lambda$, $M_0^\lambda$ is the restriction of $M_0$ to the set of places $P_\lambda$, $D_o^\lambda$ is the submatrix of the incidence matrix $D$ that has rows that correspond to the places in $P_\lambda$ and columns that correspond to the transitions in $T_o$, $D_{uo}^\lambda$ is the submatrix of the incidence matrix $D$ that has rows that correspond to the places in $P_\lambda$ and columns that correspond to the transitions in $T_\lambda$ (namely, the incidence matrix of the unobservable subnet $N_\lambda$), $\sigma_o$ is the restriction of $\sigma$ to $T_o$, $\sigma_{uo}$ is the restriction of $\sigma$ to $T_\lambda$, and $n_2 = |P_\lambda| \leq n$, where $n$ is the number of places.

Since the unobservable subnet $N_\lambda$ is structurally bounded, there exists an $n_2$-dimensional column vector $y_\lambda$ with strictly positive integer entries such that $y_\lambda^T D_{uo}^\lambda \leq 0_{l_\lambda}^T$, where $l_\lambda = |T_\lambda|$. Multiplying both sides of Eq. (3.8) by $y_\lambda^T$ on the left, we get

$$y_\lambda^T M^\lambda = y_\lambda^T M_0^\lambda + y_\lambda^T D_o^\lambda \sigma_o + y_\lambda^T D_{uo}^\lambda \sigma_{uo} \geq 0. \quad (3.9)$$

As $y_\lambda^T D_{uo}^\lambda \sigma_{uo} \leq 0$,

$$y_\lambda^T M^\lambda \leq y_\lambda^T M_0^\lambda + y_\lambda^T D_o^\lambda \sigma_o .$$

Let $c_1 = y_\lambda^T M_0^\lambda$ and $c_2$ be the maximal entry of $y_\lambda^T D_o^\lambda$. Then

$$y_\lambda^T M^\lambda \leq c_1 + c_2(1 1 \cdots 1) \sigma_o \leq c_1 + c_2 k ,$$

where $k$ is the length of the observation sequence $\omega$. As $y_\lambda$ is a vector with
strictly positive integer entries,

\[ M^\lambda(p) \leq y^T_\lambda M^\lambda \leq c_1 + c_2 k \]

for every \( p \in P_\lambda \). Combining the above result with the result in Lemma 3.1, we get the following upper bound on the number of consistent markings.

**Theorem 3.2** Consider a labeled Petri net \((G, \Sigma, L)\) with a known initial marking \(M_0\) and a structurally bounded unobservable subnet \(N_\lambda = (P_\lambda, T_\lambda, F_\lambda, W_\lambda)\); i.e., there exists an \(n_2\)-dimensional column vector \(y_\lambda\) with strictly positive integer entries that satisfies \(y^T_\lambda D^\lambda uo \leq 0^T_\lambda\). If the observation sequence \(\omega\) has length \(k\), then the number of consistent markings is upper bounded by

\[
(1 + a_1 + a_2 k)^{n-n_2}(1 + c_1 + c_2 k)^{n_2},
\]

where\(^5\) \( a_1 = \max_{p \in P \setminus P_\lambda} M_0(p), a_2 = \max_{p \in P \setminus P_\lambda, t \in T_o} D(p, t), c_1 = y^T_\lambda M_0^\lambda, \) and \(c_2\) is the maximal entry of \(y^T_\lambda D^\lambda_o\).

**Remark 3.4** If there is no unobservable transition (namely, if \(l_\lambda = 0\)), then the upper bound is exactly the bound in Lemma 3.1 because \(n_2 = |P_\lambda| = 0\). Note that the bound in Theorem 3.2 depends on the value of \(y_\lambda\) which is not unique (e.g., \(cy_\lambda\) for any positive integer \(c\) also satisfies \(cy^T_\lambda D^\lambda uo \leq 0\)); also this bound may be loose in a Petri net with a large number of places. To get a better bound, we may want to optimize \(y_\lambda\) such that \(1 + c_1 + c_2 k\) is minimized for values of \(k\) of interest; however, even if we know that a Petri net is structurally bounded, it may not be easy to find \(y_\lambda\). For some Petri nets, structural boundedness can be established; for example, input dominant Petri nets\(^6\) are easy to identify by checking conditions on each transition separately and have been shown to be structurally bounded with \(y = 1_n\) in [37]. For more discussions on how to determine if a Petri net is structurally bounded, refer to [38]. The important observation here is that for Petri nets with structurally bounded unobservable subnets, \(c_1\) and \(c_2\) are constants. \(\blacksquare\)

\(^5\)If \(P_\lambda = P\), both \(a_1\) and \(a_2\) can be taken to be 0. If \(T_\lambda = \emptyset, y_\lambda, c_1\) and \(c_2\) are not defined; as the value of \(y_\lambda\) is not important in this case, we can take both \(c_1\) and \(c_2\) to be 0 (this case also extends to Proposition 3.4 and Corollary 3.1).

\(^6\)A Petri net \(G\) is called an *input dominant Petri net* if for each transition \(t \in T\), the sum of the input arc weights is larger than or equal to the sum of the output arc weights [37].
A special case of structurally bounded unobservable subnets is the class of unobservable subnets for which there exists an \( n_2 \)-dimensional column vector \( y_\lambda \) with strictly positive integer entries such that \( y_\lambda^T D_{uo} < 0_{l_\lambda}^T \); we call such Petri nets deadlock structurally bounded Petri nets.

**Definition 3.1** A Petri net \( G \) with \( n \) places and \( m \) transitions is deadlock structurally bounded if there exists an \( n \)-dimensional column vector \( y \) with strictly positive integer entries such that \( y^T D < 0^T_m \).

Note that there exist Petri nets that are structurally bounded but not deadlock structurally bounded; for example, the Petri net shown in Fig. 3.4 is structurally bounded with the vector \( y = (1 \ 1)^T \) but there is no vector \( y \) with strictly positive integer entries such that \( y^T D < 0^T_m \).

To further clarify this class of Petri nets, the following proposition indicates that an important subclass of deadlock structurally bounded Petri nets is the class of acyclic Petri nets without source transitions; the proof of the following proposition, along with the algorithm for computing a corresponding vector \( y \) in the case of acyclic Petri nets without source transitions, is provided in Appendix A.

**Proposition 3.3** If a Petri net is acyclic and has no source transitions, then the Petri net is deadlock structurally bounded.

**Proof:** Refer to Appendix A.

Now we consider Problem 3.2 if the unobservable subnet is deadlock structurally bounded. First we rewrite Eq. (3.9) as

\[
y_\lambda^T M_0^\lambda + y_\lambda^T D_o^\lambda \sigma_o \geq -y_\lambda^T D_{uo}^\lambda \sigma_{uo}.
\]
If the unobservable subnet is deadlock structurally bounded, then
\[
c_1 + c_2 k \geq y_\lambda^T M_0^\lambda + y_\lambda^T D_0^\lambda \sigma_o \\
\geq -y_\lambda^T D_{uo}^\lambda \sigma_{uo} = \sum_{t \in T_{\lambda}} \sigma(t),
\]
where \( c_1 = y_\lambda^T M_0^\lambda \) and \( c_2 \) is the maximal entry of \( y_\lambda^T D_0^\lambda \). Therefore, every \( \sigma \) must satisfy
\[
\sum_{t \in T_{\lambda}} \sigma(t) \leq c_1 + c_2 k .
\] (3.10)

Now we can generalize the result in Proposition 3.2 to handle unobservable transitions. To simplify the representation, we assume that

- the empty label is denoted by \( \lambda \) and there are \( d \) other labels \( e_1, e_2, \ldots, e_d \), where \( 1 \leq d \leq m \);
- \( l_{\lambda} := |T_{\lambda}|, \ l_{e_i} := |T_{e_i}| \geq 2 \) for \( i = 1, \ldots, j \) while \( l_{e_i} := |T_{e_i}| = 1 \) for \( i = j + 1, \ldots, d \), where \( 0 \leq j \leq d \);
- the observation is a sequence of labels \( \omega \) of length \( k \geq 0 \), in which \( e_i \) appears \( k_{e_i} \geq 0 \) times and \( k_{e_1} + \ldots + k_{e_d} = k \).

**Proposition 3.4** Consider a labeled Petri net \((G, \Sigma, L)\) with a known initial marking \( M_0 \) and a deadlock structurally bounded unobservable subnet \( N_{\lambda} = (P_{\lambda}, T_{\lambda}, F_{\lambda}, W_{\lambda}) \); i.e., there exists an \( n_2 \)-dimensional column vector \( y_{\lambda} \) with strictly positive integer entries that satisfies \( y_\lambda^T D_{uo}^\lambda < 0^T_{\lambda} \). If the observation is a sequence of labels \( \omega \) of length \( k \geq 0 \), in which \( e_i \) appears \( k_{e_i} \geq 0 \) times and \( k_{e_1} + \ldots + k_{e_d} = k \), then the number of consistent markings is upper bounded by
\[
\left( \frac{c_1 + c_2 k + l_{\lambda}}{l_{\lambda}} \right) \times \prod_{i=1}^{j} \left( \frac{k_{e_i} + l_{e_i} - 1}{l_{e_i} - 1} \right),
\] (3.11)
where \( c_1 = y_\lambda^T M_0^\lambda \) and \( c_2 \) is the maximal entry of \( y_\lambda^T D_0^\lambda \). The upper bound is defined to be \( \left( \frac{c_1 + c_2 k + l_{\lambda}}{l_{\lambda}} \right) \) if \( j = 0 \).

**Proof:** Given an observed sequence of labels \( \omega \), a firing vector \( \sigma \) corresponding to some marking \( M \in \mathcal{C}(\omega) \) satisfies not only the equation array in Eq. (3.2) but also the inequality Eq. (3.10). Following the argument in the
proof for Proposition 3.2, we can show that there are \((c_1 + c_2 k + l_\lambda)\) possible combinations of \(\{\sigma(t) | t \in T_\lambda\}\) such that Eq. (3.10) holds. Then, the number of firing vectors (and therefore an upper bound on the number of consistent markings) is upper bounded by

\[
\left(c_1 + c_2 k + l_\lambda \right) \times \prod_{i=1}^{j} \left( k_{e_i} + l_{e_i} - 1 \right)
\]

because the transition sets \(T_{e_1}, T_{e_2}, \ldots, T_{e_j}, T_\lambda\) are pairwise disjoint. Note that if there is no unobservable transition (namely, if \(l_\lambda = 0\)), then the upper bound is exactly the bound in Proposition 3.2.

**Corollary 3.1** Consider a labeled Petri net \((G, \Sigma, L)\) with a known initial marking \(M_0\) and a deadlock structurally bounded unobservable subnet \(N_\lambda = (P_\lambda, T_\lambda, F_\lambda, W_\lambda)\); i.e., there exists an \(n_2\)-dimensional column vector \(y_\lambda\) with strictly positive integer entries that satisfies \(y_\lambda^T D_{uo}^\lambda < 0_{l_\lambda}^T\). If the observation sequence \(\omega\) has length \(k\), then the number of consistent markings is upper bounded by

\[
\frac{k \prod_{i=1}^{j} (l_{e_i} - 1)^{j-1}}{((l - 1)!)^j} \times \left( c_1 + c_2 k + l_\lambda \right) \tag{3.12}
\]

where \(j \geq 1\) is the number of nondeterministic labels, \(\tilde{l} = \max\{l_{e_1}, l_{e_2}, \ldots, l_{e_j}\}\), \(l = \min\{l_{e_1}, l_{e_2}, \ldots, l_{e_j}\}\), \(c_1 = y_\lambda^T M_0^\lambda\) and \(c_2\) is the maximal entry of \(y_\lambda^T D_{uo}^\lambda\). If \(j = 0\), we define the value of Eq. (3.12) to be \((c_1 + c_2 k + l_\lambda)\).

**Proof:** Direct application of Theorem 3.1 to Proposition 3.4.

**Remark 3.5** The four bounds we have described in this section are all polynomial in the length of the observation sequence. The bounds can be classified into two categories:

- bounds with complexity \(\mathcal{O}(k^n)\) obtained by bounding the number of tokens in every place (refer to Lemma 3.1 and Theorem 3.2),

- bounds with complexity \(\mathcal{O}(k^{j(l-1)})\) or \(\mathcal{O}(k^{j(l-1) + l_\lambda})\) obtained by bounding the number of firing vectors (refer to Theorem 3.1 and Corollary 3.1).

We investigate the application of such bounds to the state estimation problem in the next section.
3.5 Implications for State Estimation

In this section, we first review earlier work on state estimation in the context of DESs modeled by Petri nets, and then apply the results in Sections 3.3 and 3.4 to establish that state estimation problems in labeled Petri nets are solvable with algorithms that have computational complexity polynomial in the length of the observation sequence.

State estimation in DESs has been considered extensively [23,29,34,35,39]. For instance, using the theory of generalized state space systems, the authors of [39] proposed an extended reduced Luenberger observer to reconstruct the Petri net marking and the firing vector based on partially measured places and transitions. In [29], Giua et al. studied the problem of estimating the marking of a Petri net based on observations of transition firings, under the assumption that the net structure is known. Even though the initial marking may be totally or partially unknown, they were able to obtain a marking estimate that is a lower bound of the actual marking. In [23], Giua et al. studied state estimation in $\lambda$-free labeled Petri nets in which transitions can share labels (namely, transitions are not necessarily deterministic); they showed that, as long as nondeterministic transitions are contact-free, the set of consistent markings can be represented by a linear system with a fixed structure which does not depend on the length of the observation sequence. In [35], the authors considered Petri nets with deterministic and unobservable transitions under the assumption that the unobservable subnet is acyclic and backward conflict-free, and used a linear system with a fixed structure to represent markings that are consistent with a given observation sequence. The backward conflict-free assumption is relaxed in [34] and, as a result, the representation of consistent markings becomes more complicated. The major disadvantages of previous approaches are restrictive assumptions on the unobservable subnet (e.g., the acyclic assumption, the backward conflict-free assumption) and the labeling function (e.g., the contact-free assumption).

Using the polynomial bounds in Section 3.3 and Section 3.4, we show that the state estimation problem can be solved in a more general setting with reasonable complexity.

7Nondeterministic transitions are contact-free if any pair of nondeterministic transitions $t_i$ and $t_j$ does not share input or output places (i.e., $t_i^f \cap t_j^f = \emptyset$, where $t^f = t^u \cup t^s$) [23].

8The backward conflict-free assumption means that all unobservable transitions have no common output places.
The problem we consider here is the estimation of consistent markings when we observe a sequence of labels in labeled Petri nets that may have nondeterministic transitions and/or unobservable transitions. We first focus on Petri nets without unobservable transitions (namely, \( \lambda \)-free labeled Petri nets).

### 3.5.1 State Estimation: Petri Nets without Unobservable Transitions

Given the observation of a sequence of labels \( \omega \) of length \( k \) (corresponding to an unknown underlying firing sequence \( t_{s_1}t_{s_2}\cdots t_{s_k} \)), we are interested in computing all consistent markings under the assumptions that

- **A1** the structure of the Petri net \( G \) is known,
- **A2** the initial marking \( M_0 \) is known, and
- **A3** the labeling function is \( \lambda \)-free (i.e., all transition firings are associated with labels that can be observed).

We first recall Algorithm 4 in [23], as shown in Algorithm 1. The idea of Algorithm 1 is quite simple: for each observed label, one considers all transitions that share the label and are enabled; then, one simply enumerates all consistent markings. Note that at Line 11, \( C(\omega_i) \) gives the set of all markings that are consistent with the observation sequence \( \omega_i \).

To analyze the computational complexity of the above algorithm, assume that the number of consistent markings is \( N_{k-1} \) (or \( N_k \)) when the observation sequence has length \( k - 1 \) (or \( k \)). If a new event \( e \) is observed after the observation sequence \( \omega_{k-1} \), then we need to

1. consider every transition associated with \( e \) for every consistent marking in \( C(\omega_{k-1}) \),
2. obtain the next marking if it is enabled,
3. compare the new marking with other consistent markings computed at stage \( k \),
4. add it to \( C(\omega_k) \) if it is not already included.
Algorithm 1 Computation of Consistent Markings in $\lambda$-free Labeled Petri Nets

**Input:** A $\lambda$-free labeled Petri net $(G, \Sigma, L)$ and a streaming observation sequence $\omega$

**Output:** $C(\omega)$ at each step

1. Initialize $\omega_0 = \lambda$, $C(\omega_0) = \{M_0\}$;
2. Let $i = 0$;
3. Wait until a new event $e$ is observed;
4. Let $i = i + 1$, $\omega_i = \omega_{i-1}e$, $C(\omega_i) = \emptyset$;
5. for any $M \in C(\omega_{i-1})$ do
6.   for any $t$ such that $L(t) = e$ and $M[t]$ do
7.     Compute $M' = M + D(\cdot, t)$;
8.     Set $C(\omega_i) = C(\omega_i) \cup \{M'\}$;
9.   end for
10. end for
11. Output $C(\omega_i)$;

Roughly, the complexity is $N_{k-1} \times |T_e| \times (n + n + nN_k)$ in terms of scalar comparisons and additions, where the first $n$ is the number of comparisons to determine whether some transition associated with $e$ is enabled, the second $n$ is the number of additions to compute the next marking, and $nN_k$ is the number of comparisons\(^9\) to check whether the next marking has already been added into $C(\omega_k)$. Using the result in Theorem 3.1, it is easy to verify that the complexity of one step computation from $k - 1$ to $k$ is\(^{10}\) $O(n\bar{k}^{2j(l-1)})$. Therefore, the complexity of computing $C(\omega_k)$ starting from $M_0$ using Algorithm 1 can be bounded by $O(n\bar{k}^{2j(l-1)+1})$. This means that we can compute all consistent markings with complexity that is polynomial in the length of the observation sequence, even for general Petri nets without the need for any particular assumption on the labeling function (such as the contact-free assumption in [23]).

\(^9\)Note that $nN_k$ is the number of comparisons needed using linear search but it can be improved using more sophisticated methods (e.g., binary search [40]).

\(^{10}\)In reality the number of tokens in each place at time step $k$ can be $1 + a_1 + a_2k$ in the worst case (as shown in Lemma 3.1), which means that the complexity of each addition or comparison is $\log_2(1 + a_1 + a_2k)$ in terms of bit operations. Therefore, the complexity of one step computation from $k - 1$ to $k$ is $N_{k-1} \times |T_e| \times (n + n + nN_k) \times \log_2(1 + a_1 + a_2k)$, i.e., $O(n\bar{k}^{2j(l-1)} \log k)$. Since $\log k < k$ for any positive integer $k$, the one step complexity can be bounded by $O(n\bar{k}^{2j(l-1)+1})$. Similarly, the complexity of computing $C(\omega_k)$ starting from $M_0$ can be changed accordingly but it is still polynomial in $k$. 

34
Figure 3.5: Plot of three quantities of interest against the length of the observed sequence of labels. 1 (green curve): number of consistent markings; 2 (blue curve): number of firing vectors; 3 (pink curve): upper bound obtained from Proposition 3.2.

**Remark 3.6** The computational complexity of Algorithm 1 is an exponential function of the structural parameters $j$ and $\overline{l}$; that is to say, the time required for computing consistent markings increases exponentially in the structural parameters (though it does increase polynomially in the length of the observation sequence). Note that this is also true for marking estimation in Petri nets under the contact-free assumption on nondeterministic transitions (see Proposition 10 of [23]). Also note that the bound on the computational complexity of Algorithm 1 is not exponential in the number of transitions but exponential in parameters $j$ and $\overline{l}$, which characterize the distribution of transitions with respect to labels.

**Example 3.2** Consider the Petri net structure on the left of Fig. 2.2 with the initial marking $M_0 = (100 100 2)^T$ and the labeling function $L$ satisfying $L(t_1) = L(t_4) = L(t_5) = a$, $L(t_2) = L(t_3) = b$ and $L(t_6) = c$ (note that both $M_0$ and $L$ are different from the ones in Example 2.2). We randomly generate\footnote{Labels $a$, $b$ and $c$ are chosen with probabilities $P(a) = \frac{3}{6}$, $P(b) = \frac{2}{6}$ and $P(c) = \frac{1}{6}$.} a sequence of labels $\omega = caabaaacbabaaabbbbaabbababa$
of length 30 and then compute consistent markings using Algorithm 1. The number of consistent markings, the number of firing vectors, and the upper bound obtained in Proposition 3.2 are plotted against the length of this observation sequence in Fig. 3.5. Note that the number of firing sequences is $1.4599 \times 10^{11}$, which is much larger than 1989 (namely, the upper bound in Proposition 3.2); the actual number of consistent markings is even lower at 1027. Figure 3.6 shows a plot of the maximal number of consistent markings over 100 random observation sequences of length 30 together with the upper bound obtained from Theorem 3.1. Note that in this example, since $j = 2$, $\ell = 3$ and $l = 2$, the bound on the number of consistent markings is $(\frac{k}{2} + \frac{3}{2})^4$, i.e., $\mathcal{O}(k^4)$. Since the Petri net is easily verified to be 212-bounded, the upper bound in Theorem 3.1 may exceed for large $k$ the number of all possible markings (i.e., $213^4 \approx 2.06 \times 10^9$). In the scenario we considered, the upper bound at length 30 is $7.412 \times 10^4$, which is much smaller than the number of all possible markings. Clearly, for bounded Petri nets, the polynomial bound is useful in applications where the state of the system becomes periodically known or is periodically reset (so that $k$ remains effectively bounded).

Remark 3.7 Figures 3.5 and 3.6 suggest that the upper bounds obtained in Proposition 3.2 and Theorem 3.1 can be much larger than the number of consistent markings. The tightness of the bounds, however, depends on the structure of the labeled Petri net and the observation sequence. In fact, the upper bound in Proposition 3.2 can be reached in Petri nets in which

(i) the incidence matrix has full column rank (because in such cases, given an initial marking, any consistent marking corresponds to a unique firing vector), and

(ii) the initial marking is large enough or there are enough source transitions (so that there exists at least one firing sequence for every possible firing vector).

The upper bound obtained in Theorem 3.1 is close if (in addition to the conditions mentioned above to reach the upper bound in Proposition 3.2), the following conditions hold:

Note that in this case there is no risk of generating a sequence of observed labels whose set of consistent markings is empty since the initial marking has been chosen large enough. Also note that in reality the sequence of labels observed will be the result of transition activity in the Petri net.
Figure 3.6: Plot of the maximal number of consistent markings over 100 randomly generated observation sequences together with the polynomial upper bound of Theorem 3.1. 1 (green curve): maximal number of consistent markings; 2 (red curve): upper bound obtained from Theorem 3.1.

(a) There are no deterministic transitions (so that the inequality in Eq. (3.7) becomes equality).

(b) \( T = \underline{t} \) (so that the inequality in Eq. (3.4) becomes equality).

(c) The number of observed labels for every nondeterministic label is the same (so that the inequality in Eq. (3.5) becomes equality).

For example, the number of markings that are consistent with the observation sequence \( \omega = e^k \) for the Petri net in Fig. 3.1 is \( \binom{k+2}{2} \) since the incidence matrix has full column rank and there are enough source transitions; in this case, the bound in Theorem 3.1, i.e.,

\[
\frac{(k + \frac{3}{2})^2}{2} = \binom{k + 2}{2} + \frac{1}{8},
\]

is almost identical to the number of consistent markings because conditions (a), (b) and (c) are all satisfied.
Algorithm 2 Computation of Consistent Markings in General Labeled Petri Nets

**Input:** A labeled Petri net \((G, \Sigma, L)\) and a streaming observation sequence \(\omega\)

**Output:** \(C(\omega)\) at each step

1: Initialize \(\omega_0 = \lambda\), \(C(\omega_0) = UR(M_0)\);
2: Let \(i = 0\);
3: Wait until a new event \(e\) is observed;
4: Let \(i = i + 1\), \(\omega_i = \omega_{i-1}e\), \(C(\omega_i) = \emptyset\);
5: for any \(M \in C(\omega_{i-1})\) do
6: for any \(t\) such that \(L(t) = e\) and \(M[t]\) do
7: Compute \(M' = M + D(:, t)\);
8: Set \(C(\omega_i) = C(\omega_i) \cup \{M'\}\);
9: end for
10: end for
11: Let \(C' = \bigcup_{M \in C(\omega_i)} UR(M)\), \(C(\omega_i) = C'\);
12: Output \(C(\omega_i)\);
13: Goto 3.

3.5.2 State Estimation: Petri Nets with Unobservable Transitions

In this subsection we relax Assumption A3 and allow the existence of unobservable transitions (i.e., \(\lambda\)-labeled transitions) in the Petri net. More formally, we keep Assumptions A1 and A2 but replace Assumption A3 of Section 3.5.1 with the assumption below:

A3’ labels associated with firings of transitions in \(T \setminus T_\lambda\) can be observed but transitions in \(T_\lambda\) cannot (they are associated with the empty label).

First we introduce the unobservable reach from a marking \(M\), which is slightly modified from the one in [33].

**Definition 3.2** Given a labeled Petri net \((G, \Sigma, L)\) and a marking \(M\), the unobservable reach \(UR(M)\) from \(M\) is \(\{M' \mid \exists S \in T_\lambda^*, M[S]M'\}\).

To compute the set of consistent markings when unobservable transitions are present, we can modify Algorithm 1 slightly using the notion of the unobservable reach. Notice that Algorithm 2 assumes that the unobservable reach from a reachable marking \(M\) can be computed (e.g., it is finite).
In general, the computational complexity of Algorithm 2 can be high since the method essentially amounts to reachability analysis. However, if the unobservable subnet is structurally bounded, the computational complexity of Algorithm 2 is also polynomial in the length of the observation sequence; this can be easily argued using the result in Theorem 3.2.

**Example 3.3** Consider the Petri net in Fig. 3.7, in which the labeling function is $L(t_1) = a$ and $L(t_2) = \lambda$. For this Petri net, we can choose $y_\lambda = 1$, and therefore $c_1 = 0$ and $c_2 = 1$. If the observation sequence is $\omega = a^k$, the bounds given by both Theorem 3.2 (note that $a_1 = a_2 = 0$) and Corollary 3.1 are $k+1$, which is also the exact number of markings that are consistent with $\omega$. This example shows that the bounds in Theorem 3.2 and Corollary 3.1 are tight for certain types of Petri nets.

3.6 Implications for Fault Diagnosis

In this section, we apply Corollary 3.1 to fault diagnosis problems in Petri nets, obtaining algorithms which have computational complexity that is polynomial in the length of the observation sequence.

3.6.1 Introduction

As systems modeled by DESs become more complicated, failures appear more often and consequences become potentially severe; as a result, fault diagnosis has emerged as an extremely important task in many applications.

One of the most extensively studied fault models is the one where faults are modeled as unobservable events, or unobservable state transitions. Following this fault model, the authors of [31] focused on DESs modeled by finite state machines, introduced the notions of fault types and diagnosability, and designed fault diagnosers to test for diagnosability and implement online
fault diagnosis. Later on, their work was extended to DESs modeled by Petri nets [33, 41, 42]. Specifically, the authors of [41] constructed fault diagnosers solely based on marking variations in observed places while the authors of [42] constructed fault diagnosers based on not only marking variations but also observed transitions; they followed an approach similar to the one in [31] so that previous results could be directly applied. In [33], a distributed version of the diagnoser approach by Sampath et al. (see [31]) was proposed for place-bordered Petri nets. In contrast to these direct applications of the diagnoser approach of [31], Giua et al. constructed basis reachability trees for bounded labeled Petri nets (based on the notions of basis markings and justifications) so that faults can be detected [34]. In [19], minimal diagnosers (that use observations from a minimum number of observable places) are constructed to immediately detect and isolate the firing of fault transitions. There are also other fault diagnosis methods (for different fault models) that are based on algebraic coding techniques [43, 44], net unfolding techniques [45, 46], interpreted Petri net formulations [47], and others.

In the fault diagnosis problem we consider, the system is modeled as a labeled Petri net, and faults are modeled as unobservable state transitions that may be classified into one or more types. We construct an online monitor to compute the belief we have regarding the occurrence of each fault type. Given a sequence of observations, the belief of a particular fault type is a measure of our confidence regarding the occurrence of faults of that type and is defined as the ratio of the sum of the weights of possible paths\textsuperscript{12} that contain that fault type to the sum of the weights of all possible paths. The weight of a path can be viewed as a measure of its likelihood and can be a function of the transitions involved, the times at which they occur or other factors. When a path’s weight is taken to be the product of the weights of its individual transitions, we show that the proposed monitor can be implemented recursively with complexity that is polynomial in the length of the observed sequence of labels. The approach can be generalized to deal with partially observed Petri nets (for details, refer to [11]).

\textsuperscript{12}Possible paths are sequences of transitions that are consistent with a given sequence of observations.
3.6.2 Problem Formulation

As mentioned earlier, faults are modeled as unobservable transitions and are partitioned into \( q \) types \( \Delta_F = \{F_1, F_2, \cdots, F_q\} \). Let \( T_F \) be the set of fault transitions, and \( T_{F_i} \) be the set of fault transitions whose type is \( F_i \). Then we have (i) \( T_F \subseteq T_{uo} \); (ii) \( T_F = T_{F_1} \cup T_{F_2} \cup \cdots \cup T_{F_q} \); (iii) \( T_{F_i} \cap T_{F_j} = \emptyset \) if \( i, j \in \{1, 2, \ldots, q\} \) and \( i \neq j \).

In many existing diagnosis approaches (e.g. [31, 34]), when there are many execution paths that are consistent with a given observation sequence, the diagnosis result provides very coarse information. For example, in [31], there are three types of labels that can be associated with each system state that is consistent with the observation sequence: label ‘A’ is used when there is ambiguity about the occurrence of certain faults; label ‘N’ is used when there is no fault in any of the consistent paths reaching that state; label ‘\( F_i \)’ means that a fault belonging to type ‘\( F_i \)’ has occurred. Inspired by the notion of “belief” proposed in [48] in probabilistic inference settings, we attempt to capture how confident we are about the occurrence of faults of certain fault types, based on the observations seen so far, by defining a suitable measure. This measure is called belief and is a way of capturing the likelihood of different execution paths (e.g., via probabilities, power consumption, or other constraints).

To introduce the belief measure, we assume that for any firing sequence \( S \) enabled at \( M_0 \), there exists a positive weight function \( wt(M_0, S) \) which captures the likelihood of the sequence \( S \). We first introduce a notion of consistent markings which is slightly different from Definition 2.10.

**Definition 3.3** Given a labeled Petri net \((G, \Sigma, L)\) with initial marking \( M_0 \) and an observed label sequence \( \omega \in \Sigma^* \), the set of consistent firing sequences is defined as

\[
S(\omega) = \{S \mid S \in T^*T_o : M_0[S] \text{ and } L(S) = \omega\}
\]

if \( \omega \neq \lambda \), and \( S(\lambda) = \emptyset \) otherwise; the set of consistent markings is defined as

\[
C(\omega) = \{M \in N_0^o \mid \exists S \in S(\omega) : M_0[S]M\}
\]

if \( \omega \neq \lambda \), and \( C(\lambda) = \{M_0\} \) otherwise, where \( T_o = T\backslash T_\lambda \) and \( T^*T_o \) denotes the concatenation of \( T^* \) and \( T_o \).
Definition 3.4 Given a labeled Petri net \((G, \Sigma, L)\) with the observation sequence \(\omega\) and partition \(\Delta_F = \{F_1, F_2, \cdots, F_q\}\) of fault transitions \(T_F\), the belief on the occurrence of faults belonging to type \(F_i\) is

\[
b(\omega, F_i) = \frac{\sum_{S \in S(\omega) \text{ and } \exists t \in T_{F_i} \text{ appearing in } S} wt(M_0, S)}{\sum_{S \in S(\omega)} wt(M_0, S)}; \quad (3.13)
\]

the belief on the normal operation of the system is

\[
b(\omega, N) = \frac{\sum_{S \in S(\omega) \text{ and no fault appears in } S} wt(M_0, S)}{\sum_{S \in S(\omega)} wt(M_0, S)}. \quad (3.14)
\]

In other words, the belief \(b(\omega, F_i)\) is the ratio of the sum of path likelihoods that contain a fault belonging to type \(F_i\) to the sum of all path likelihoods that are consistent with \(\omega\).

- If \(b(\omega, F_i) = 0\), then no fault of type \(F_i\) can possibly have occurred.
- If \(b(\omega, F_i) = 1\), then a fault of type \(F_i\) must have occurred.
- If \(0 < b(\omega, F_i) < 1\), then there is ambiguity about the occurrence of faults of type \(F_i\).

Therefore, the diagnosis results in [31] can also be obtained via the use of the belief measure as a special case; moreover, if \(b(\omega, F_i)\) is closer to 1 (or 0), we are more (or less) confident about the occurrence of a fault of type \(F_i\).

Now we formulate the fault diagnosis problem. Given a labeled Petri net \((G, \Sigma, L)\) with partition \(\Delta_F = \{F_1, F_2, \cdots, F_q\}\) of fault transitions \(T_F\), a weight function \(wt(M_0, S)\) defined for any firing sequence \(S\) from \(M_0\), and an observation sequence \(\omega\) (due to an underlying unknown firing sequence \(S\) such that \(\omega = L(S)\)), our goal is to calculate the beliefs on the occurrence of each fault type, i.e., \(b(\omega, F_i)\) for \(1 \leq i \leq q\), and the belief on the normal running of the system, i.e., \(b(\omega, N)\).

3.6.3 Online Monitor for Belief Calculation

In this subsection we describe cases when we can efficiently calculate beliefs in a recursive manner. The resulting diagnoser can be used online to determine
the beliefs associated with each fault type given the observation sequence seen so far.

To systematically define the weight function \( wt(M_0, S) \) from each individual transition in \( S \) and be able to calculate the belief efficiently in a recursive online manner, we assume the existence of a weight function

\[
wt(M, t) : R(G, M_0) \times T_M \rightarrow R_0^+, \]

where \( T_M \) denotes the set of transitions that are enabled at marking \( M \) and \( R_0^+ \) denotes the set of nonnegative real numbers. Such a weight function can describe the likelihood of transition \( t \) at marking \( M \) (e.g., it can capture the probability that a particular transition occurs at a particular state [49]), or it can correspond to the cost of transition \( t \)’s firing at marking \( M \) (which is a generalization of the cost function in [9]). We can then define the weight function for a sequence of transitions \( S = t_s_1 t_s_2 \cdots t_s_k \) that is enabled at marking \( M_0 \) (i.e., \( M_0[t_s_1] M_1[t_s_2] M_2 \cdots [t_s_k] M_k \)) as an extension of \( wt(M, t) \):

\[
wt(M_0, S) = wt(M_0, t_s_1) \otimes wt(M_1, t_s_2) \otimes \cdots \otimes wt(M_{k-1}, t_s_k),
\]

where \( \otimes \) is some associative abstract operation. The exact operation \( \otimes \) depends on the meaning of the \( wt(M, t) \). For example, \( \otimes \) can be \( \times \), the ordinary product of real numbers, if \( wt(M, t) \) captures the probability of the occurrence of \( t \) at \( M \) (and transition firings at different markings are independent); and \( \otimes \) can be \( + \), the ordinary sum of real numbers, if \( wt(M, t) \) captures the cost of firing \( t \) at \( M \); other choices are also possible. To simplify the discussion, we use \( \times \) from now on.

Two interesting choices for the weight function \( wt(M, t) \) are the following. The first choice is \( wt(M, t) = 1 \) for any \( M \) and any \( t \) that is enabled at \( M \). With this choice, \( wt(M_0, S) = 1 \) for any firing sequence \( S \). Therefore, \( b(\omega, F_i) \) is the ratio of the number of paths that contain a fault belonging to type \( F_i \) to the number of all paths that are consistent with \( \omega \). The second interesting choice for the weight function is \( wt(M, t) = \frac{1}{|T_M|} \), which allows each transition enabled at \( M \) to occur with equal weight \( \frac{1}{|T_M|} \). This latter weight function gives more weight to paths that have smaller branching. Other reasonable weight functions are also possible (e.g., weight functions might be defined based on probabilistic Petri nets by randomizing choices [49]).
Example 3.4 In Fig. 3.8, the observation is simply the label $a$; markings are denoted using black dots (markings at the lowest level form the consistent markings), observable (or unobservable) transitions are denoted using solid (dashed) lines; there is only one fault transition $t_f$ of type $F$. If $wt(M, t) = 1$, then $b(a, F) = \frac{1}{5}$; however, if $wt(M, t) = \frac{1}{|T_M|}$, then $b(a, F) = \frac{1}{3}$. ■

Now we make the following reasonable assumption to ensure that there is no arbitrarily long sequence of unobservable transitions; this is also a requirement in [31]. This assumption also guarantees that the online monitor we introduce in this section can be constructed efficiently.

Assumption 3.1 The unobservable subnet of the given labeled Petri net is deadlock structurally bounded.

Remark 3.8 For a deadlock structurally bounded Petri net, there exists vector $y$ with strictly positive integer entries such that $y^T D < 0_m^T$. Left-multiplying $y^T$ on both sides of the state equation (2.1), we get

$$y^T M = y^T M_0 + y^T D \sigma.$$ 

Since $y^T D < 0_m^T$, the firing of any transition $t$ will strictly decrease the value of $y^T M_0$ to get $y^T M$, but $y^T M$ is lower bounded by 0. Therefore, the length of any firing sequence must be finite. In other words, there exists no firing sequence of infinite length in deadlock structurally bounded Petri nets. ■

The other assumption below is on the weight function $wt(M_0, S)$.

Assumption 3.2 The path weight function $wt(M_0, S)$ for a path

$$M_0 | t_{s_1} | M_1 \cdots | t_{s_k} | M_k$$
Equation (3.13) can be rewritten as

\[
b(\omega, F_i) = \frac{\sum_{M \in \mathcal{C}(\omega)} \left( \sum_{S \in \mathcal{S}(\omega), \text{ } M_0[S>M, \text{ and } \exists t \in T_{F_i} \text{ appearing in } S} \text{wt}(M_0, S) \right)}{\sum_{M \in \mathcal{C}(\omega)} \left( \sum_{S \in \mathcal{S}(\omega) \text{ and } M_0[S>M} \text{wt}(M_0, S) \right)}
\]

by grouping the sum of \( \text{wt}(M_0, S) \) for sequences that lead to the same consistent marking \( M \). Note that

\[
\sum_{S \in \mathcal{S}(\omega), \text{ } M_0[S>M, \text{ and } \exists t \in T_{F_i} \text{ appearing in } S} \text{wt}(M_0, S)
\]

is completely determined by the fault type \( F_i \) and the marking \( M \), and

\[
\sum_{S \in \mathcal{S}(\omega) \text{ and } M_0[S>M} \text{wt}(M_0, S)
\]

is completely determined by the marking \( M \). Therefore, given an observation sequence \( \omega \), we can use the following data structure as a node to represent each consistent marking \( M \in \mathcal{C}(\omega) \) and to also include information on fault occurrences: \( (M, K) \), where (i) \( M \) is a consistent marking in \( \mathcal{C}(\omega) \); (ii) \( K \) is a \((q + 2)\)-dimensional row vector in which

\[
K(i) = \sum_{S \in \mathcal{S}(\omega), \text{ } M_0[S>M, \text{ and } \exists t \in T_{F_i} \text{ appearing in } S} \text{wt}(M_0, S) \quad \text{for } i = 1, 2, \ldots, q,
\]

i.e., the weighted sum of consistent paths that drive the system from \( M_0 \) to \( M \) and also contain faults belonging to type \( F_i \);

\[
K(q + 1) = \sum_{S \in \mathcal{S}(\omega) \text{ and } M_0[S>M} \text{wt}(M_0, S),
\]

i.e., the weighted sum of all consistent paths that drive the system from \( M_0 \)
Figure 3.9: Calculation of \( C_{\text{ext}}(\omega e) = \{ (M_9, K_9), (M_{10}, K_{10}) \} \) from \( C_{\text{ext}}(\omega) = \{ (M_1, K_1), (M_2, K_2), (M_3, K_3), (M_4, K_4) \} \).

\[
K(q+2) = \sum_{S \in S(\omega), \text{ } M_0 \text{ } \mid S \text{ } > \text{ } M, \text{ } \text{and no fault appears in } S} \text{wt}(M_0, S),
\]

i.e., the weighted sum of consistent paths that drive the system from \( M_0 \) to \( M \) without faults. The \( K(q+2) \) entry takes Eq. (3.14) into account.

We define \( C_{\text{ext}}(\omega) = \cup_{M \in C(\omega)} \{ (M, K) \} \). Using the data structure \( (M, K) \) in \( C_{\text{ext}}(\omega) \), the beliefs can be computed using the following equations:

\[
b(\omega, F_i) = \frac{\sum_{(M, K) \in C_{\text{ext}}(\omega)} K(i)}{\sum_{(M, K) \in C_{\text{ext}}(\omega)} K(q+1)}, \quad i = 1, 2, \ldots, q,
\]

\[
b(\omega, N) = \frac{\sum_{(M, K) \in C_{\text{ext}}(\omega)} K(q+2)}{\sum_{(M, K) \in C_{\text{ext}}(\omega)} K(q+1)}.
\]

Note that \( C_{\text{ext}}(\omega) \) are essentially consistent markings plus weight sums for belief calculation. We can compute consistent markings recursively as shown in Section 3.5 and we now show that we can also compute \( C_{\text{ext}}(\omega) \) recursively, which implies that beliefs can be computed recursively.

If \( \omega = \lambda \), there is only one consistent marking \( M_0 \). Therefore, \( C_{\text{ext}}(\lambda) \) is initialized as \( \{ (M_0, K_0) \} \), where \( K_0 = [0_{1 \times q}, 1] \). Now suppose we have computed \( C_{\text{ext}}(\omega) \) and we observe a new label \( e \); we need to calculate \( C_{\text{ext}}(\omega e) \).

A typical plot of the process is shown in Fig. 3.9. In this figure, \( C_{\text{ext}}(\omega) = \{ (M_1, K_1), (M_2, K_2), (M_3, K_3), (M_4, K_4) \} \), \( C_{\text{ext}}(\omega e) = \{ (M_9, K_9), (M_{10}, K_{10}) \} \), dashed lines denote unobservable transitions, solid lines denote observable transitions that generate \( e \), \( t_{f_1} \) and \( t_{f_2} \) are fault transitions, and other nodes are intermediate nodes. There cannot be cycles in the update process because of Assumption 3.1.
We divide the computation of $C_{\text{ext}}(\omega e)$ from $C_{\text{ext}}(\omega)$ into two steps: in the first step, we ignore the observed label $e$ and obtain all markings reachable by firing only unobservable transition sequences (e.g., the graph consisting of dashed edges and related nodes in Fig. 3.9); in the second step, we take the label $e$ into account to get $C_{\text{ext}}(\omega e)$ (e.g., the graph consisting of solid edges and related nodes in Fig. 3.9).

In the first step, we use $C_{\text{uo}}$ to denote the set of nodes $(M, K)$ such that $(M, K)$ is reachable from some node in $C_{\text{ext}}(\omega)$ by firing one or more unobservable transitions. Therefore, $C_{\text{ext}}(\omega) \subseteq C_{\text{uo}}$. When updating beliefs given the current node $(M, K) \in C_{\text{uo}}$ and one enabled (unobservable) transition $t$, the state $M'$ in the next node $(M', K')$ can be obtained as $M' = M + D(\cdot; t)$. Depending on whether $t$ is a fault transition and on whether $M'$ has appeared in some node computed in the current step, there are four cases to consider:

- **Case I**: $t$ is a normal transition (enabled at $M$ in $(M, K)$) and $M'$ does not exist in nodes computed at the current step (e.g., transition $t_1$ is enabled at $M_1$ and the resulting marking $M_5$ has not been computed, as shown in Fig. 3.9). In this case, create a new node $(M', K')$ (e.g., $(M_5, K_5)$), in which $K'(i) = K(i) \times wt(M, t)$ for $i = 1, \ldots, q + 2$. The reason this works is because, if $S_i$ denotes the set of firing sequences that drive the system from $M_0$ to $M$ and satisfy corresponding properties for different $i$'s, then $K'(i) = \sum_{S \in S_i} wt(M_0, S) = \sum_{S \in S_i} (wt(M_0, S) \times wt(M, t)) = wt(M, t) \times \sum_{S \in S_i} wt(M_0, S) = wt(M, t) \times K(i)$ for $i = 1, 2, \ldots, q + 2$.

- **Case II**: $t$ is a fault transition (enabled at $M$ in $(M, K)$) belonging to type $F_i$ and $M'$ does not exist in nodes computed at the current step (e.g., fault transition $t_{f1}$ is enabled at $M_5$ and the resulting marking $M_6$ has not been computed, as shown in Fig. 3.9). In this case, create a new node $(M', K')$ (e.g., $(M_6, K_6)$), in which $K'(j) = K(j) \times wt(M, t)$ for $j \in \{1, \ldots, q+1\} - \{i\}$, $K'(i) = K(q+1) \times wt(M, t)$ and $K'(q+2) = 0$.

- **Case III**: $t$ is a normal transition (enabled at $M$ in $(M, K)$) and $M'$ exists in node $(M', K')$ computed at the current step (e.g., transition $t_2$ is enabled at $M_7$ and the resulting marking $M_6$ was computed from $(M_5, K_5)$ by firing transition $t_{f2}$, as shown in Fig. 3.9). In this case, let $K'(i) = K'(i) + K(i) \times wt(M, t)$ for $i = 1, \ldots, q + 2$. 
• Case IV: \( t \) is a fault transition (enabled at \( M \) in \( (M, K) \)) belonging to type \( F_i \) and \( M' \) exists in node \( (M', K') \) computed at the current step (e.g., fault transition \( t_{f2} \) is enabled at \( M_3 \) and the resulting marking \( M_7 \) was computed from \( (M_2, K_2) \) by firing transition \( t_3 \), as shown in Fig. 3.9). In this case, let \( K'(j) = K'(j) + K(j) \times wt(M, t) \) for \( j \in \{1, \ldots, q + 1\} - \{i\} \), \( K'(i) = K'(i) + K(q + 1) \times wt(M, t) \). Note that in this case \( K'(q + 2) \) has no change.

The argument for Case II is similar to Case I except that \( t \) is a fault transition of type \( F_i \). In Case IV (Case III is similar to Case IV and we do not discuss it explicitly here), \( (M_7, K_7) \) has already been created based on the firing of \( t_3 \) from \( (M_2, K_2) \). As the firing of fault transition \( t_{f2} \) from \( (M_3, K_3) \) also results in the marking \( M_7 \), we need to update \( K_7 \). The value of \( K_7(q + 2) \) does not change because it denotes the weighted sum of paths without faults; for \( j \in \{1, \ldots, q + 1\} - \{i\} \), \( K_7(j) = K_7(j) + K_3(j) \times wt(M_3, t_{f2}) \) while \( K_7(i) = K_7(i) + K_3(q + 1) \times wt(M_3, t_{f2}) \) because all paths from the initial marking to \( M_7 \) going through \( M_3 \) contain the fault transition \( t_{f2} \) of type \( F_i \).

Note that there is a dependency issue in this update process: more specifically, suppose we calculate \( (M, K) \) in Fig. 3.9 by considering all firing sequences consisting of unobservable transitions from \( (M_1, K_1) \), \( (M_2, K_2) \), \( (M_3, K_3) \), \( (M_4, K_4) \) sequentially. After we are done with \( (M_1, K_1) \) and \( (M_2, K_2) \), we consider \( (M_3, K_3) \) and update \( (M_7, K_7) \). However, \( (M_6, K_6) \) also depends on \( (M_7, K_7) \), and therefore we also need to update all markings that are reachable from \( M_7 \). To avoid this dependency issue, we can first generate the reachability graph in Fig. 3.9 and then update the value \( K \) in node \( (M, K) \) only when all nodes \( (M', K') \) (where \( M \) is reachable from \( M' \)) have been updated. Though the computation of \( C_{uo} \) amounts to reachability analysis in general, Assumption 3.1 ensures that \( |C_{uo}| \) is finite and that the reachability graph is acyclic (because there exists no firing sequence of infinite length in deadlock structurally bounded Petri nets; for more details, refer to Remark 3.8).

In the second step, we take the label \( e \) into account to get \( C_{ext}(\omega e) \). What we need to do is just to consider all transitions that are mapped to \( e \) at all markings in \( C_{uo} \). The update process is essentially the same as Cases I and III discussed in the calculation of \( C_{uo} \) (because no fault transitions can be mapped to label \( e \)).
Algorithm 3 Online Monitor

Input: A labeled Petri net \((G, \Sigma, L)\) (satisfying Assumptions 3.1 and 3.2) with partition \(\Delta_F = \{F_1, F_2, \ldots, F_q\}\) of fault transitions \(T_F\) and a streaming sequence of observed labels \(\omega\)

Output: Output the belief

1: Initialize \(\omega_0 = \lambda, C_{\text{ext}}(\omega_0) = \{(M_0, K_0)\}\);
2: Let \(i = 0\);
3: Wait until a new label \(e\) is observed;
4: Let \(i = i + 1, \omega_i = \omega_{i-1}e, C_{\text{ext}}(\omega_i) = \emptyset\);
5: Calculate \(C_{\text{uo}}\) following the rules in Cases I-IV
6: for any \((M, K) \in C_{\text{uo}}\) do
7: for any \(t\) such that \(L(t) = e\) and \(M[t]\) do
8: Compute \(M' = M + D(\cdot, t)\)
9: if \(M'\) does not appear in any node of \(C_{\text{ext}}(\omega_i)\) then
10: Calculate \(K'\) using the rule in Case I and add \((M', K')\) into \(C_{\text{ext}}(\omega_i)\);
11: end if
12: if \(M'\) exists in node \((M', K')\) of \(C_{\text{ext}}(\omega_i)\) then
13: Update \(K'\) using the rule in Case III;
14: end if
15: end for
16: end for
17: Output the belief \(b(\omega_i, F_i)\) for \(i = 1, \ldots, q\) and \(b(\omega_i, N)\) using Eq. (3.15);
18: Goto 3.

The complete description of the online monitor is given in Algorithm 3.

We now briefly explain the algorithm. Lines 1 and 2 initialize \(C_{\text{ext}}(\omega)\); Lines 3-18 update \(C_{\text{ext}}(\omega)\) as an extra label \(e\) is observed. In Line 5, we compute all nodes in \(C_{\text{uo}}\) that can be reached from consistent nodes in \(C_{\text{ext}}(\omega_{i-1})\) by firing sequences consisting of unobservable transitions, while in Lines 6-16, we update \(C_{\text{uo}}\) to compute \(C_{\text{ext}}(\omega_i)\) by considering all transitions that can be mapped to the most recently observed label \(e\). This is in line with the definition of the set of consistent firing sequences in that the last transition is required to be observable. In practical applications, a threshold value \(\alpha\) can be selected so that we can declare a fault of type \(F_i\) has occurred if \(b(\omega, F_i) > \alpha\). If \(\alpha\) is too close to 0, there can be many false alarms; on the other hand, if \(\alpha\) is too close to 1, there can be many missed detections. Therefore, \(\alpha\) needs to be adjusted to reflect the real system based on long-term observations and expert knowledge.
The above algorithm is based on consistent markings that can be reached via different firing sequences. We next argue that the complexity of Algorithm 3 is polynomial in the length $k$ of the observed label sequence.

Here we use $C'(\omega)$ to denote the set of markings consistent with $\omega$ as defined in Definition 2.10. Essentially, in this definition the last transition in the firing sequence $S$ (such that $M_0[S]M$ and $L(S) = \omega$) need not be observable. Therefore, $C(\omega) \subseteq C'(\omega)$ and $|C(\omega)| \leq |C'(\omega)|$. In Corollary 3.1, we show that $|C'(\omega)|$ is $O(k^b)$ for a given labeled Petri net with a deadlock structurally bounded unobservable subnet, where $k$ is the length of $\omega$, $b = j(\bar{l} - 1) + l_\lambda$, and $l_\lambda = |T_\lambda|$. Thus, $|C(\omega)|$ is also $O(k^b)$.

Suppose we have computed $C_{\text{ext}}(\omega_{k-1})$. In the first step of computing $C_{\text{ext}}(\omega_k)$, we first compute $C_{\text{uo}}$. As $|C_{\text{uo}}| \leq |C'(\omega_{k-1})|$, $|C_{\text{uo}}|$ is $O((k - 1)^b)$, which implies $C_{\text{uo}}$ can be calculated with complexity polynomial in $k$. In the second step, if a new label $e$ is observed after the observation sequence $\omega_{k-1}$, then we need to

(i) consider every transition associated with $e$ for every marking in $C_{\text{uo}},$

(ii) obtain the next marking $M$ if it is enabled,

(iii) compare the new marking $M$ with other consistent markings computed at stage $k$, and calculate/update the corresponding $K$,

(iv) add $(M, K)$ to $C_{\text{ext}}(\omega_k)$ if it is not already included or update $K$ if $M$ is already included.

Roughly, the complexity is $|C_{\text{uo}}| \times |T_e| \times (n + n + nN_k + q + 2)$ in terms of scalar comparisons and additions, where the first $n$ is the number of comparisons to determine whether some transition associated with $e$ is enabled, the second $n$ is the number of additions to compute the next marking, $nN_k$ is the number of comparisons (using linear search) to check whether the next marking has already been added into $C_{\text{ext}}(\omega_k)$ ($N_k := |C_{\text{ext}}(\omega_k)|$), and $q + 2$ is the number of additions to update $K$. Using the previous bound, it is easy to verify that the complexity of the second step is $O((\bar{l}(k - 1)^b(nk^b + q))$. Clearly, the computation of $C_{\text{ext}}(\omega_k)$ from $C_{\text{ext}}(\omega_{k-1})$ is polynomial in $k$, which implies that the complexity of computing $C_{\text{ext}}(\omega_k)$ starting from $(M_0, K_0)$ using Algorithm 3 is also polynomial in $k$. 

50
3.6.4 Extensions

In this subsection, we discuss possible extensions of the monitoring scheme to handle repeated faults and multiple faults.

In some systems the same type of faults might repeat multiple times (e.g., intermittent or non-persistent faults may occur repeatedly) [50]. There can also be multiple faults. In [43], faults are modeled as token corruptions in places; in that setting, tokens in multiple places can be corrupted because of multiple faults. In Petri nets allowing concurrency, multiple faults can occur simultaneously when faults are modeled as unobservable transitions. However, in the fault diagnosis problem we consider, at most one transition can fire at any instant and there is only one (unknown) underlying firing sequence. Therefore, repeated faults are defined as faults of the same type occurring in the same firing sequence while multiple faults are defined as faults of different types occurring in the same firing sequence.

For example, we consider two types of faults $T_{F_1}$ and $T_{F_2}$. Given an observation sequence $\omega$, we say faults in $T_{F_1}$ and $T_{F_2}$ possibly occur if there is a firing sequence $S \in S(\omega)$ such that there exist transitions $t_1 \in T_{F_1}$ and $t_2 \in T_{F_2}$ and $t_1, t_2$ appear in $S$. To calculate $b(\omega, F_1, F_2)$, i.e., the belief on the occurrence of faults of types $F_1$ and $F_2$, we need to expand the $K$ vector with an additional entry $K(q + 3)$ to track the sum of the weights of firing sequences containing faults of these two types. Moreover, we need to remember if a fault of type $F_1$ (or $F_2$) has occurred in some sequence so that we can update the entry $K(q + 3)$ when we find another fault of type $F_2$ (or $F_1$) later on. Except for the need to remember the occurrence of faults of a single type, the rules for the update are essentially the same as those in Cases I-IV. We can calculate beliefs on the occurrence of repeated faults and faults of other multiple fault types in a similar manner.

3.7 Implications for Reachability Checking

In this section, we show that the reachability problem for certain classes of Petri nets can be solved using finite enumeration based on Proposition 3.1.
3.7.1 Introduction

Given a Petri net, the reachability problem asks if a final target marking is reachable from a known initial marking. The problem plays a central role in Petri net theory as a lot of other problems (e.g., liveness analysis and deadlock checking) are recursively equivalent to it [51]. In fault diagnosis applications (e.g., [31, 42]), the system model is usually assumed to be deadlock-free (i.e., for any reachable state, there has to exist at least one state transition that is possible); therefore, the reachability problem is important for checking if such fault diagnosis methods can be applied.

Though the reachability problem has been shown to be decidable [52], its complexity is still an open problem. There have been tremendous efforts toward solving the reachability problem both for general Petri nets and subclasses of Petri nets (e.g., [5, 53–55]). For example, the reachability tree method is proposed in [53] to address the problem but there can be loss of information due to the introduction of $\omega$-markings. In [5], reachability results are summarized for several classes of Petri nets, such as acyclic Petri nets, marked graphs, and others. In [54], the reachability problem for Petri nets without transition invariants is studied; however, the efficiency and complexity of the proposed method is unclear. In [55], the authors propose a net transformation procedure that converts a general Petri net into an acyclic Petri net (in order to utilize the sufficient condition for reachability of acyclic Petri nets in [5]), but it is unclear how to compute (or bound) the number of stages for net expansion.

In this section, we study the reachability problem for a class of Petri nets, called strictly monotone Petri nets (refer to Definition 3.5). For such Petri nets, the reachability problem can be solved by searching the states that can be reached via firing sequences of length less than or equal to a calculable constant $k$. The bound on the length of firing sequences can be used to analyze the complexity of the reachability algorithm and can also be used as (an upper bound on) the number of stages for net expansion in [55] (as the number of stages required to solve the reachability problem is equal to the length of a certain firing sequence that reaches the final target marking). The algorithm is important and useful in many applications, including fault diagnosis. For example, the algorithm can be used to determine if some faulty or undesirable states are possible given a sequence of observations in a
labeled Petri net (for details, refer to [14]). A generalization of the algorithm to monotone Petri nets is given in [14].

### 3.7.2 Problem Formulation

We first introduce the class of Petri nets, called strictly monotone Petri nets.

**Definition 3.5** A Petri net $G$ is said to be *strictly monotone* if there exists an $n$-dimensional real vector $y$ such that $y^T D > 0_m$.

Strictly monotone Petri nets generalize deadlock structurally bounded Petri nets (refer to Definition 3.1) because they allow the vector $y$ to have negative entries.

**Remark 3.9** There are no transition invariants in strictly monotone Petri nets. This can be argued by contradiction: suppose there exists a nonzero vector $X$ with nonnegative integer entries satisfying $DX = 0_n$; then, on the one hand, $y^T DX = y^T(DX) = 0$, whereas on the other hand, $y^T DX = (y^T D)X > 0$.

Now we formulate the following reachability problem.

**Problem 3.3** Given a strictly monotone Petri net $G = \langle N, M_0 \rangle$ and a target marking $M$, is $M$ reachable from $M_0$?

### 3.7.3 Reachability Algorithm

In this subsection, we analyze the reachability problem for strictly monotone Petri nets and show that it can be solved by finite enumeration.

We first derive a necessary condition for a marking to be reachable. Given a monotone Petri net $G$ and an initial marking $M_0$, suppose $y$ is a real vector satisfying $y^T D > 0_m$ and marking $M$ is reachable from $M_0$ through a firing sequence $S$ which maps to the firing vector $\sigma$. If we multiply on the left by $y^T$ on both sides of the state equation $M = M_0 + D\sigma$, we get

$$y^T M = y^T M_0 + y^T D\sigma$$

and then

$$y^T (M - M_0) = y^T D\sigma .$$

53
Algorithm 4 Reachability Algorithm for Strictly Monotone Petri Nets

**Input:** Petri net $G = \langle N, M_0 \rangle$ and a given target marking $M$

**Output:** $M$ is reachable or unreachable

1: Check if $G$ is strictly monotone. If not, exit without output.
2: Calculate $y$ such that $y^T D > 0^T_m$. Then calculate $l$ using Eq. (3.16) and $k$ using Eq. (3.17). If $k \leq 0$, exit with $M$ unreachable.
3: Enumerate all markings that are reachable by a firing sequence of length less than or equal to $k$ and check if $M$ is one of them (this checking is needed only when the firing sequence is longer than or equal to $l$): if $M$ is one of these markings, then $M$ is reachable; otherwise, $M$ is unreachable.

Letting $Z = y^T D$, we have

$$(\min_i Z(i)) \times \sum_{t \in T} \sigma(t) \leq y^T D \sigma \leq (\max_i Z(i)) \times \sum_{t \in T} \sigma(t).$$

Therefore, $l \leq \sum_{t \in T} \sigma(t) \leq k$, where\(^{13}\)

$$l = \left\lceil \frac{y^T (M - M_0)}{\max_i Z(i)} \right\rceil$$ (3.16)

and

$$k = \left\lfloor \frac{y^T (M - M_0)}{\min_i Z(i)} \right\rfloor.$$ (3.17)

Note that $l$ and $k$ are well defined because $Z(i) > 0$ for any $1 \leq i \leq m$.

Because $\sum_{t \in T} \sigma(t)$ is the length of the firing sequence $S$, we know that if the marking $M$ is reachable, it must be reachable by a firing sequence $S$ of length $l \leq |S| \leq k$, where $|S|$ denotes the length of the firing sequence $S$. In other words, if $M$ is not reachable by a transition sequence $S$ with length $l \leq |S| \leq k$, it is not reachable. Based on this idea, we have Algorithm 4 for reachability checking.

Step 1 and the calculation of vector $y$ (at Step 2) can be done using linear programming techniques as will be discussed shortly. At Step 2, $k$ and $l$ can be non-positive; if $k \leq 0$, it means that the marking is unreachable because $M$ is different from $M_0$ (if $M$ is the same as $M_0$, the reachability problem is

\(^{13}\)The ceiling function $\lceil x \rceil$ is the smallest integer which is larger than or equal to $x$; the floor function $\lfloor x \rfloor$ is the largest integer which is smaller than or equal to $x$. 54
trivial). The algorithm is not sensitive to the final target marking $M$ unlike some existing methods (e.g., the methods in [52, 54]) in which, if the final target marking is changed, the methodology needs to be applied from scratch.

Checking for Strict Monotonicity in Petri Nets

To check if a given Petri net is strictly monotone, we need to check if there exists a real vector $y$ such that $y^T D > 0^T_m$ and this can be done by linear programming [56]. We use $C^T y$ as the objective function, where $C$ is a nonzero vector (e.g., $C$ can be the vector with all entries being 1). Then a linear programming problem can be formulated as follows:

$$\min \ C^T y$$
$$s.t. \ y^T D > 0^T_m.$$

If there is a solution to this linear programming problem, then the Petri net is strictly monotone; otherwise, it is not. As the linear programming problem can be solved with polynomial complexity, checking the strictly monotone property can be done efficiently.

In Algorithm 4, we need to search all markings reachable by firing sequences of length up to $k$. Therefore, one natural idea for the objective function is to minimize $k$ when we choose the vector $y$. That is to say, the programming problem can be formulated as follows:

$$\min \ \left[ \frac{y^T(M - M_0)}{\min_i Z(i)} \right]$$
$$s.t. \ y^T D > 0^T_m;$$

where $Z = y^T D$. If the programming problem (which is now nonlinear) has a solution, then the Petri net is strictly monotone and also the solution $y$ will minimize the search depth $k$. An example that demonstrates this approach is given in Section 3.7.4.

Computational Complexity

Steps 1 and 2 of Algorithm 4 involve a linear programming problem, and
can be solved with polynomial complexity [56]. To analyze the complexity of Step 3, we recall (and rephrase) Proposition 3.1 as below.

**Proposition 3.5** Consider a Petri net $G$ with a known initial marking $M_0$ and transitions $t_1, \ldots, t_m$ sharing a single label $e$. If we observe an observation sequence $\omega = ee \cdots ee$ of length $k$, then the number of consistent markings is upper bounded by $\binom{k+m-1}{m-1}$, where $m \geq 1$ and $k \geq 0$.

In other words, the number of states reachable by a firing sequence of length $k$ is bounded by $\binom{k+m-1}{m-1}$, i.e., $O(k^{m-1})$. We denote the number of states reachable by firing sequences of length $k - 1$ (or $k$) as $N_{k-1}$ (or $N_k$). Given we have calculated all states reachable by firing sequences of length $k - 1$, to calculate all states reachable by firing sequences of length $k$, we need to

(i) consider each transition for every state at step $k - 1$,

(ii) obtain the next state if this transition is enabled,

(iii) compare the new state with other states already computed at step $k$ using linear search,

(iv) add it to states reachable at step $k$ if it is not already present.

Roughly, the complexity is $N_{k-1} \times m \times (n + n + nN_k)$ in terms of scalar comparisons and additions, where the first $n$ is the number of comparisons to determine whether some transition is enabled, the second $n$ is the number of additions to compute the next state, and $nN_k$ is the number of comparisons (using linear search) to check whether the next state has already been added. Therefore, the complexity of one step computation from $k - 1$ to $k$ is $O(nmk^{2(m-1)})$ and the complexity of computing all states reachable from $M_0$ by firing sequences of length up to $k$ is $O(nmk^{2m-1})$. This is also the complexity of Step 3 as the marking $M$ can be unreachable, which implies that we have to generate all possible markings up to step $k$. The complexity result can be improved using search algorithms that are more advanced than linear search, such as binary search, hash functions, or others.
3.7.4 Example

To illustrate Algorithm 4, we consider the Petri net in Fig. 3.10 with initial marking \( M_0 = (2 \ 0)^T \) and incidence matrix

\[
D = \begin{bmatrix}
2 & -1 \\
-1 & 1 \\
\end{bmatrix}.
\]

The Petri net is verified to be strictly monotone using the vector \( y = (0.5 \ 0.8)^T \). First, we check if the marking \( M_1 = (1 \ 0)^T \) is reachable or not. As \( Z = y^T D = (0.2 \ 0.3) \), we have

\[
k = \left\lfloor \frac{y^T (M_1 - M_0)}{\min_i Z(i)} \right\rfloor = \left\lfloor \frac{-0.5}{0.2} \right\rfloor = -3.
\]

As \( k \leq 0 \), marking \( M_1 \) is unreachable. Indeed, \( M_1 \) is unreachable because the token sum in \( M_1 \) is 1, which violates the fact that the token sum of places \( p_1 \) and \( p_2 \) must be larger than or equal to 2 for all reachable markings.

Now we check if the marking \( M_2 = (7 \ 0)^T \) is reachable or not. Using Eq. (3.16) for calculating \( l \) and Eq. (3.17) for calculating \( k \), we have \( k = 12 \) and \( l = 9 \). By running Algorithm 4, \( M_2 \) is verified to be reachable with a firing sequence of length 10, which lies in the set \( \{l, l+1, ..., k\} \) as expected.

Now we use the nonlinear programming formulation in the previous subsection to minimize the search length \( k \). The nonlinear programming problem is

\[
\min \left\lfloor \frac{y^T (M_2 - M_0)}{\min_i Z(i)} \right\rfloor \\
\text{s.t. } y^T D > 0^T_m,
\]

Figure 3.10: A simple strictly monotone Petri net.
where $Z = y^T D$. Using the nonlinear programming solver `fmincon` in Matlab with the initial point $y_0 = (0.5 \ 0.8)^T$, we find one solution $y = (0.2 \ 0.3)^T$. With this $y$, the search length is $k = 10$ (and also $l = 10$); in other words, we only need to check markings that are reachable with a firing sequence of length 10. The reachability result is the same as in the previous case which uses $y = (0.5 \ 0.8)^T$, but this time less running time is needed (as $k$ is 10 instead of 12).

3.8 Summary

In this chapter, we present upper bounds on the number of consistent markings in DESs when the underlying model is a labeled Petri net. For $\lambda$-free labeled Petri nets, we showed that the number of consistent markings is at most polynomial in the length of the observation sequence. Polynomial bounds on the number of consistent markings were also obtained for Petri nets with structurally bounded unobservable subnets. Note that if the Petri net is bounded, then eventually the number of consistent markings is bounded by the size of the reachability set. In this case, if our bound exceeds the size of the reachability set for some (large) value of $k$, we should use the size of the reachability set as the bound on the number of consistent markings.

Our bounds imply that the online state estimation problem can be solved efficiently with complexity that is polynomial in the length of the observation sequence in a very general setting. Note that in many supervisory control and fault diagnosis applications (that require state estimation), the Petri net model of the plant is fixed and only the number of observed labels increases with time. Therefore, broadly speaking, our bounds can also be used to argue the computational complexity of supervisory control and fault diagnosis algorithms, as long as the labeled Petri net model satisfies the assumptions in this paper. For example, the polynomial bounds are used in the computational complexity analysis for least-cost firing sequence estimation [9], fault diagnosis (as shown in Section 3.6; for more details, refer to [11]), reachability checking (as shown in Section 3.7; for more details, refer to [14]), and supervisor synthesis. Here is a very brief summary of the supervisor synthesis problem (for details, refer to [13]):

- The system is modeled as a partially observed and partially controlled
Petri net\textsuperscript{14} $Q$ with labeling function $L$.

- The goal is to synthesize a maximally permissive control policy such that the system is guaranteed to avoid entrance to any state within $M_F$, a given finite set of forbidden states.

- The algorithm is an online algorithm which first calculates an estimate of current system states (due to limited sensing capability) and then synthesizes a control policy to avoid forbidden states.

- The online algorithm is shown to have computational complexity that is polynomial in the length of the observation sequence by applying Theorem 3.2.

Our bounds can also be used to guide the design of sensor configurations on observable transitions (in the form of transition labels) without having to explicitly evaluate the actual performance of sensor configurations. For example, if we assign transition labels in a way that minimizes $j(\bar{l} - 1)$, then the (bound on the) number of consistent markings at each step is also minimized according to the result in Theorem 3.1 and Corollary 3.1. In other words, the (bound on the) uncertainty introduced by the state estimation process is reduced so that we can potentially perform supervisory control and fault diagnosis more effectively.

\textsuperscript{14}The partially observed and partially controlled Petri net used here is the one in Definition 2.5 with $P_s = \emptyset$, i.e., there is no place sensor available.
CHAPTER 4

SENSOR SELECTION FOR STRUCTURAL OBSERVABILITY

In this chapter, we consider the selection of sensors so that the state of a given partially observed Petri net can be determined uniquely based on sensor information.

4.1 Introduction

Applications that involve monitoring and controlling of discrete event systems rely on information conveyed by various types of sensors that are available in the system. Usually it is impossible or undesirable to place sensors everywhere because sensors may be unavailable or prohibitively expensive for certain state transitions or other tasks. Therefore, selecting a minimum number of sensors (or a set of sensors of minimal cost) that also meets the system design requirements is critical and often mandatory.

Optimal sensor selection problems have been studied extensively in discrete event systems that can be modeled as finite state machines, e.g., [57–60]. In [57], a sequence of tests is provided to obtain a set of sensors that has minimal cost and ensures a given property (such as diagnosability). In [58], the problem of obtaining an optimal sensor configuration of minimum cardinality is shown to be computationally hard (by showing that the corresponding decision problem is \( \mathcal{NP} \)-complete) for several properties, including diagnosability, normality, and observability. The authors of [59] discuss the problem of sensor selection to achieve observability with minimum cost and show that polynomial time algorithms to find good\(^1\) approximate solutions to this problem most likely do not exist (at least under certain complexity assumptions). Minimal\(^2\) sensor selection to fulfill a desired formal property

\(^1\)An approximate solution is “good” if its total cost is very close to the optimal cost. For the precise meaning, refer to [59].

\(^2\)A sensor configuration is *minimal* if it is a minimal element in a partially ordered
is shown to be generally \(NP\)-hard in [60]; for properties that have mask-monotonic behavior (e.g., (co-)observability, normality, state-observability, and diagnosability), “top-down” and “bottom-up” methods that have polynomial complexity and achieve a minimal sensor configuration are proposed in [60].

In this chapter, we focus on sensor selection in DESs that can be modeled as Petri nets. There is only limited previous work on sensor selection problems when the underlying model is a Petri net. For example, in [61], observability notions based on inputs and outputs are used as criteria when optimizing the selection of sensors in interpreted Petri net models; in this case, genetic algorithms are used to approximate the optimal sensor selection, but the method only applies to bounded Petri nets and the proposed algorithm converges slowly to a suboptimal solution.

In the sensor selection problem we consider, we formulate the notion of structural observability, i.e., the ability to uniquely determine the system state at any given time step\(^3\) based on sensor information up to that time step, knowledge of the system model, and an arbitrary but known initial state. The requirement for uniquely determining the system state at any given time step is motivated by a number of applications where complete knowledge of the system state is absolutely necessary. Examples include the following:

- In supervisory control, control policies for a large number of synthesis methods (e.g., [25, 62–68]) are defined as a function from any reachable state to a control action. Implicitly, these methods require that the system state at any given time step is exactly known. There is only a limited number of supervisor synthesis methods that are based on state estimates (e.g., [12, 29, 69, 70]) because such control policies are difficult to formulate in an optimal manner [29]. The problem is that algorithms based on state estimates may also be forced to prevent transition firings that lead the system from one admissible state to another admissible state. More importantly, the use of state estimates may significantly reduce the performance of the closed-loop system and, in particular, it

---

\(^3\) Time steps refer to the times at which transitions in a firing sequence fire. For example, time step \(i\) refers to the time at which the \(i\)-th transition in a firing sequence fires.
may lead to a deadlock [30].

- In the design of safety-critical systems, discrete event systems which model user-interfaces must be immediately observable in order to have good properties, i.e., interfaces need to accurately represent the underlying system to the user, so that the user will not be misled or confused [71].

- In system monitoring, a unique (possibly corrupted) state is required to perform fault detection and correction [43, 44].

Structural observability requires that the current system state is determined uniquely without delay for any arbitrary but known initial state. As shown later in this chapter (and proved in Appendix B), even if one allows a finite delay in the definition of structural observability, the requirements for the two notions (non-delayed and delayed structural observability) remain essentially the same.

After we formulate and analyze structural observability, we consider the placement of a minimum number of sensors in the system in order to enforce this property. Unlike sensor selection problems for DESs modeled by finite automata, we allow two types of sensors (in order to model both place and transition observability): place sensors indicate the number of tokens in a particular place (e.g., vision sensors), and transition sensors indicate the firing of a transition in a given subset of transitions (e.g., motion sensors). To simplify the problem and gain a better understanding of it, we consider two subproblems: the optimal place sensor selection (OPSS) problem and the optimal transition sensor selection (OTSS) problem. We first establish that the OPSS problem is computationally hard by showing that the corresponding decision problem is $\mathcal{NP}$-complete via a polynomial reduction from the minimum vertex cover problem (refer to Problem 2.1) to this decision problem. We also show that the OPSS problem can be reduced with polynomial complexity to the binary integer programming problem (refer to Problem 2.2) which can be solved optimally using existing binary integer programming solvers (at least for small problem instances). As an alternative to the binary integer programming based approach, we also propose four approximation algorithms to approach the optimal solution. Unlike the OPSS problem, the OTSS problem can be solved efficiently in time that is
polynomial in the number of places and transitions. Sensor selection problems (with constraints on the way transitions might share sensors) are also considered.

In the next section, we formulate the optimal sensor selection problems to achieve structural observability. In Section 4.3, we characterize structural observability and give existence conditions (for both the OPSS and OTSS problems) that can be checked with polynomial complexity, i.e., algorithm complexity that is polynomial in the number of places and transitions of the Petri net. In Section 4.4, we show that the OPSS problem is \(\mathcal{NP}\)-complete by reducing the minimum vertex cover problem to it. The OPSS problem can be solved by transforming it into the binary integer programming problem (as shown in Section 4.5) or using approximation algorithms (four such algorithms are proposed in Sections 4.6 and 4.7). In Section 4.8, we show that the OTSS problem can be solved efficiently with polynomial complexity. In Section 4.9, we consider two sensor selection problems that take constraints (in the way transitions can share sensors) into account. Section 4.10 summarizes this chapter.

### 4.2 Problem Formulation

Before we formulate sensor selection problems, we first use the following example to illustrate the basic idea.

**Example 4.1** For the net on the right of Fig. 2.3, consider \(V = (1 \ 1 \ 1)\) and \(L\) defined as \(L(t_1) = a\), \(L(t_2) = L(t_3) = b\), \(L(t_4) = c\) and \(L(t_5) = \lambda\). Suppose \(M_0 = (2 \ 0 \ 1 \ 0)^T\) and the firing sequence \(t_3t_5\) occurs. Then, the system trajectory is

\[
M_0[t_3]M_1[t_5]M_2,
\]

where \(M_1 = (0 \ 0 \ 2 \ 0)^T\) and \(M_2 = (0 \ 0 \ 1 \ 1)^T\). Given the place sensor configuration \(V\) and the labeling function \(L\), the available sensor information is

\[
(2 \ 0 \ 1)^T \rightarrow b \rightarrow (0 \ 0 \ 2)^T \rightarrow (0 \ 0 \ 1)^T.
\]

Though no label is observed when the system evolves from \(M_1\) to \(M_2\), we can infer that unobservable transition \(t_5\) has occurred from the token change in place \(p_3\), because only the firing of \(t_5\) can decrease the token number in \(p_3\)
The above example shows how the information from place sensors can be used to reduce uncertainty due to transition labels; on the other hand, the information from transition labels can also be used to reduce uncertainty due to place sensors. Given a partially observed Petri net, the general sensor selection problem consists of choosing a place sensor configuration $V$ and a labeling function $L$ such that $||V|| + |\Sigma|$ is minimized and the system state can be determined uniquely based on sensing information, knowledge of the system model, and an arbitrary but known initial state. Note that we focus mainly on minimizing the total number of sensors but, more generally, the cost of sensors could be different resulting in uneven weighting and changing the problem to that of minimizing the total cost of all sensors; these extensions are discussed in Chapter 5.2.

We now formalize the concept of structural observability which ensures that the system state can be determined uniquely given a set of place sensors and transition sensors.

**Definition 4.1** Given a place sensor configuration $V$ and a labeling function $L$, a partially observed Petri net $Q$ is **structurally observable** if for an arbitrary but known initial state $M_0$ and any firing sequence from $M_0$, the system state $M$ at any given time step can be determined uniquely based on observations from place sensors and transition sensors up to that time step.

The notion of structural observability requires that the current system state is determined uniquely without delay for an arbitrary but known initial marking. If one allows a finite delay in that definition, the notion of $K$-delayed structural observability for a finite nonnegative integer constant $K$ can be stated as follows, and can be shown to be equivalent to structural observability (refer to Appendix B).

**Definition 4.2** Given a place sensor configuration $V$ and a labeling function $L$, a partially observed Petri net $Q$ is **$K$-delayed structurally observable** if for an arbitrary but known initial state $M_0$ and any firing sequence from $M_0$, the system state $M$ at any given time step $i \geq 0$ can be determined uniquely based on observations from place sensors and transition sensors no later than time step $i + K$. 

64
Remark 4.1 There are several properties that are related to, but differ from, the notion of structural observability as defined in this work. The closest one is immediate observability [71], i.e., the ability to determine the current state based only on partial information about the state and either the last or next event; the main difference between these two notions is that structural observability requires that the state can be determined for an arbitrary but known initial state. Structural marking observability [29] is defined in a different setting and has a different meaning: more specifically, in [29] each transition has a unique label and the goal is to determine if there exists an observation sequence for any unknown initial marking such that the current state can be reconstructed. In [72], observability involves not only the information from place sensors but also control inputs.

Definition 4.3 Given a partially observed Petri net \( Q \) and a fixed labeling function \( L \) (or a fixed place sensor configuration \( V \)), a place sensor configuration \( V \) (or a labeling function \( L \)) is valid if \( Q \) is structurally observable.

To simplify the sensor selection problem and gain a better understanding of it, we consider two subproblems: the optimal place sensor selection problem given a fixed labeling function, and the optimal transition sensor selection problem given a fixed place sensor configuration.

The optimal place sensor selection problem given a fixed labeling function consists of choosing, for a Petri net with a given set of transition sensors, the minimal number of observable places to put sensors on so that the system is structurally observable.

Problem 4.1 (Optimal Place Sensor Selection (OPSS)) Given a partially observed Petri net \( Q \) and a fixed labeling function \( L \), find a valid place sensor configuration \( V_{\text{min}} \) such that for any other valid place sensor configuration \( V \), \(|V_{\text{min}}| \leq |V|\).

Similarly, the optimal transition sensor selection problem given a fixed place sensor configuration consists of choosing, for a Petri net with a given set of place sensors, the minimal number of transition labels so that the system is structurally observable.

Problem 4.2 (Optimal Transition Sensor Selection (OTSS)) Given a partially observed Petri net \( Q \) and a fixed place sensor configuration \( V \), find a valid labeling function \( L_{\text{min}} : T \to \Sigma \cup \{\lambda\} \) such that \(|\Sigma|\) is minimized.
4.3 Characterization of Structural Observability

According to the state equation $M = M_0 + D\sigma$, one sufficient condition for uniquely determining the system state at any given time step is that the firing of each transition at any time step can be distinguished based on sensing information. In turn, this ensures that the firing sequence and the sequence of markings can be constructed recursively. This discussion motivates the notion of transition distinguishability.

**Definition 4.4** Given a place sensor configuration $V$ and a labeling function $L$, a partially observed Petri net $Q$ is *transition distinguishable* if, for an arbitrary but known initial state $M_0$, the firing of any of its transitions at any time step can be distinguished from any other transition firing based on observations from place sensors and transition sensors up to that time step.

**Remark 4.2** Transition distinguishability is different from event-detectability as defined in [72] in that it allows information from transition sensors (besides place sensors) to be taken into account. Another related concept is invertibility [73], i.e., the ability to reconstruct the entire event string from the observation of the output string; the main difference is that invertibility allows a finite delay and depends on the initial state.

It is clear from the above discussion that transition distinguishability is sufficient for structural observability. Transition distinguishability is also necessary if we assume that there are no identically behaving transitions in the partially observed Petri net.

**Definition 4.5** Transitions $t_1$ and $t_2$ are *identically behaving* if $D(:, t_1) = D(:, t_2)$, where $D(:, t)$ denotes the column of the incidence matrix $D$ corresponding to transition $t$.

Intuitively, the firings of identically behaving transitions drive the system from a given current state to the same resulting state; therefore, knowledge of the current and previous markings does not necessarily uniquely identify the transition (that fired among identically behaving transitions). To simplify the characterization of the conditions under which the system state can be determined uniquely, in this chapter we assume that there are no identically behaving transitions in the Petri net we study.
Remark 4.3 Identically behaving transitions need not be distinguished as far as structural observability is concerned because their firings result in the same state change. Therefore, one could simply merge identically behaving transitions as a single transition and then apply the approach proposed in this chapter.

Proposition 4.1 Given a place sensor configuration \( V \) and a labeling function \( L \), a partially observed Petri net \( Q \) is structurally observable if and only if it is transition distinguishable.

Proof: (If part) If the Petri net is transition distinguishable, then we can uniquely infer the firing sequence based on transition labels and observations from place sensors. As the initial state is known, the system state can be uniquely determined using the state equation (2.1). This process can be continued recursively for all time steps.

(Only if part) If the Petri net is not transition distinguishable, then there exists an initial marking \( M_0 \), some time step \( k \), and two transitions \( t_1 \) and \( t_2 \) such that the firings of \( t_1 \) and \( t_2 \) cannot be distinguished based on sensing information. The marking \( M \) at time step \( k \) enables both \( t_1 \) and \( t_2 \), and the firings of transitions \( t_1 \) and \( t_2 \) at marking \( M \) result in different markings as there are no identically behaving transitions. In this scenario, the system state cannot be determined uniquely and the Petri net is not structurally observable.

Proposition 4.1 implies that we can focus on the study of transition distinguishability. Given a place sensor configuration \( V \), the \(|V| \times m\) matrix \( D_V \) is constructed by keeping the rows of \( D \) that correspond to observable places with sensors. In addition, for a given labeling function \( L : T \rightarrow \Sigma \cup \{\lambda\} \), the \(|V| \times |T_e|\) matrix \( D^e_V \) is constructed for each label \( e \in \Sigma \cup \{\lambda\} \) by keeping the columns in \( D_V \) that correspond to transitions associated with label \( e \).

Proposition 4.2 Given a place sensor configuration \( V \) and a labeling function \( L \), a partially observed Petri net \( Q \) is transition distinguishable if and only if (i) for each label \( e \in \Sigma \), all columns of \( D^e_V \) are pairwise different, and (ii) for \( \lambda \), all columns of \( D^\lambda_V \) are nonzero and pairwise different.

Proof: The if part follows from the fact that for any transition \( t \), there is a unique combination of a transition label \( L(t) \) and a column vector of token changes \( D_V(:,t) \) that identifies the firing of the transition.
Now we prove the only if part by contradiction.

(i) Suppose there is a label $e \in \Sigma$ and two associated transitions $t_1$, $t_2$ such that $L(t_1) = L(t_2) = e$ and $D_V(:, t_1) = D_V(:, t_2)$. As there exists a marking $M$ under which $t_1$ and $t_2$ are both enabled, we cannot distinguish transitions $t_1$ and $t_2$ based on sensor output generated by the occurrence of either $t_1$ or $t_2$ at marking $M$ (we can always set the initial marking to be $M$); this contradicts the fact that the Petri net is transition distinguishable.

(ii) Suppose there is a transition $t$ such that $L(t) = \lambda$ and $D_V(:, t) = 0_{||V||}$, where $0_{||V||}$ is a $||V||$-dimensional column vector with all entries being 0. As there exists a marking $M$ such that $t$ is enabled, then the firing of $t$ cannot be detected based on sensor output; also a contradiction.

(iii) The case in which there are two transitions $t_1$, $t_2$ such that $L(t_1) = L(t_2) = \lambda$ and $D_V(:, t_1) = D_V(:, t_2)$ can be proved in a way similar to Case (i).

Proposition 4.3 Given a place sensor configuration $V$ and a labeling function $L$, the transition distinguishability of a partially observed Petri net $Q$ can be determined with complexity $O(nm^2)$.

Proof: For each label $e \in \Sigma \cup \{\lambda\}$ that is associated with at least 2 transitions (i.e., $|T_e| \geq 2$), we need to check whether their corresponding columns in $D_e^c$ are pairwise different. In the worst case, we need to do $||V|| \times \binom{|T_e|}{2}$ comparisons. For $\lambda$, we also need to check whether any of its transitions correspond to a zero vector in matrix $D_\lambda^c$, which needs $||V|| \times |T_\lambda|$ additional comparisons. Therefore, the total number of comparisons is

$$\#\text{comparisons} = ||V|| \times |T_\lambda| + ||V|| \times \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} \binom{|T_e|}{2}.$$

68
Let \( \sum_{e \in \Sigma \cup \{ \lambda \}} |T_e| \geq 2 \). Then

\[
\#\text{comparisons} = ||V|| \times \left( |T_\lambda| + \sum_{e \in \Sigma \cup \{ \lambda \}, |T_e| \geq 2} \frac{|T_e|^2 - |T_e|}{2} \right)
\]

\[
= ||V|| \times \left( |T_\lambda| - \frac{m_1}{2} + \sum_{e \in \Sigma \cup \{ \lambda \}, |T_e| \geq 2} \frac{|T_e|^2}{2} \right)
\]

\[
\leq ||V|| \times \left( |T_\lambda| - \frac{m_1}{2} + \frac{m_1^2}{2} \right) \tag{4.1}
\]

\[
\leq n \times \left( m + \frac{m(m - 1)}{2} \right) = \frac{nm^2 + nm}{2}, \tag{4.2}
\]

where Eq. (4.1) follows from the fact that \( \sum_{i=1}^{n} a_i^2 \leq \left( \sum_{i=1}^{n} a_i \right)^2 \) for positive \( a_i \)'s and Eq. (4.2) follows from the fact that \( ||V|| \leq n_1 \leq n \), \( |T_\lambda| \leq m \) and \( m_1 \leq m \). Therefore, transition distinguishability of a Petri net \( Q \) with outputs can be determined with complexity \( \mathcal{O}(nm^2) \), which is polynomial in the number of places \( n \) and the number of transitions \( m \). \qed

Using transition distinguishability, we can easily determine if structural observability holds for a given place sensor configuration \( V \) and a given labeling function \( L \). Now we state necessary and sufficient conditions for the existence of solutions to Problems 4.1 and 4.2. First, we define \( V_{\text{max}} = 1_{n_1} \) (where \( 1_{n_1} \) is an \( n_1 \)-dimensional column vector with all entries being 1), and define \( L_{\text{max}} \) as \( L_{\text{max}}(t) = t \) for any \( t \in T_o \) and \( L_{\text{max}}(t) = \lambda \) for any \( t \in T_{uo} \).

For the OPSS problem, if a valid place sensor configuration exists given the fixed labeling function \( L \), the problem is guaranteed to have an optimal place sensor configuration \( V_{\text{min}} \) because the number of possible place sensor configurations is finite (and equal to \( 2^{n_1} \)).

**Theorem 4.1** (Existence Condition for OPSS) Given a partially observed Petri net \( Q \) and a fixed labeling function \( L \), there exists an optimal place sensor configuration for the OPSS problem if and only if \( Q \) is transition distinguishable under \( L \) and \( V_{\text{max}} \).

**Proof:** (If part) As the Petri net is transition distinguishable under the place sensor configuration \( V_{\text{max}} \), it is structurally observable under \( V_{\text{max}} \) following Proposition 4.1, which implies that there exists at least one valid place sensor configuration. Since the total number of place sensor configurations is
finite and equal to $2^{n_1}$, the OPSS problem will have an optimal place sensor configuration.

(Only if part) If there exists an optimal place sensor configuration, we can add more sensors on the optimal place sensor configuration to get $V_{max}$, while the Petri net remains transition distinguishable following Proposition 4.2.

**Example 4.2** For the net in Fig. 4.1 with $L(t_1) = a$, $L(t_2) = L(t_3) = b$, $L(t_4) = c$ and $L(t_5) = \lambda$, we check if there exists an optimal place sensor configuration. After considering $V_{max} = 1_3$ for the net, we get the following $3 \times 5$ matrix:

$$D_{V_{max}} = \begin{bmatrix} 1 & -1 & -2 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}. $$

Following Proposition 4.2, it can be verified that the Petri net is transition distinguishable, and therefore an optimal place sensor configuration for this Petri net exists. By going through all $2^3 = 8$ possible place sensor configurations, it is easy to establish that the optimal place sensor configuration is $V_{min} = (0 \ 0 \ 1)^T$; that is to say, we only need to put a sensor on $p_3$ to complement the observation of the label sequence so that we can determine the system state uniquely.

We also have the following condition for the OTSS problem, which can be proved in a way similar to Theorem 4.1.
Theorem 4.2 (Existence Condition for OTSS) Given a partially observed Petri net \( Q \) and a fixed place sensor configuration \( V \), there exists an optimal labeling function for the OTSS problem if and only if \( Q \) is transition distinguishable under \( V \) and \( L_{\text{max}} \).

Example 4.3 We consider the Petri net in Fig. 4.2, where place \( p_4 \) is unobservable and transition \( t_5 \) is unobservable, and check if there exists an optimal labeling function given \( V = (0 \ 0 \ 1)^T \). The optimal labeling function exists following Theorem 4.2 (intuitively, the firing of unobservable transition \( t_5 \) can be identified by the token change in observable place \( p_3 \)).

Though the existence of an optimal solution to the OPSS problem can be determined with polynomial complexity, the OPSS problem itself is computationally hard (as we show in the next section, the corresponding decision problem is \( \mathcal{NP} \)-complete). On the other hand, as discussed in Section 4.8, the OTSS problem is solvable with polynomial complexity.

4.4 \( \mathcal{NP} \)-Completeness of Optimal Place Sensor Selection

In this section, we show that the OPSS problem is computationally hard by demonstrating that the corresponding decision problem is \( \mathcal{NP} \)-complete; the \( \mathcal{NP} \)-hardness is established by reducing the decision version of the minimum vertex cover problem (for details, refer to Problem 2.1) to the decision version of the OPSS problem.
Problem 4.3 (OPSS: Decision Version) Given a partially observed Petri net $Q$, a fixed labeling function $L$, and a positive integer $k \leq n$, is there a valid place sensor configuration $V'$ such that $|\ |V'|| \leq k$?

Theorem 4.3 The decision version of the OPSS problem is $\mathcal{NP}$-complete.

Proof: Problem 4.3 is shown to be $\mathcal{NP}$-complete by (i) establishing that it is in $\mathcal{NP}$ and (ii) reducing Problem 2.1 to it.

(i) Problem 4.3 is in $\mathcal{NP}$. Select a place sensor configuration $V$ such that $|\ |V|| \leq k$ and test if $V$ is valid; this test can be done with complexity that is polynomial in the number of places and transitions (see Proposition 4.3). Therefore, Problem 4.3 belongs to $\mathcal{NP}$.

(ii) Problem 4.3 is $\mathcal{NP}$-hard because Problem 2.1 can be reduced to it with polynomial complexity. Given an undirected graph $H = (Z, E)$, where $Z = \{z_1, z_2, \ldots, z_n\}$, we first rename the vertices of undirected graph $H$ to $\{1, 2, \ldots, n\}$ so that any isolated vertices have indices $n_1 + 1, \ldots, n$. We construct a partially observed Petri net $Q$ with $n$ places and $m := |E|$ transitions by making a copy of the renamed graph and attaching $p_i$ to vertex $i$ for $i = 1, 2, \ldots, n$. For every edge $e \in E$ with vertices $i, j$ in $H$, a transition $t$ is inserted in $Q$ and connected to places $p_i$ and $p_j$. We can assign directions to these arcs arbitrarily, with the constraint that the two arcs associated with the same edge in $H$ have the same direction (see Fig. 4.3). Furthermore, for each vertex, we assign different weights to each incoming (or outgoing) arc; these weights range from 1 to the number of incoming (or outgoing) arcs (which is of course bounded by $|E|$). Places $p_1, \ldots, p_{n_1}$ (or $p_{n_1+1}, \ldots, p_n$) are taken to be observable (or unobservable). Transitions are named as $t_1, t_2, \ldots, t_m$ and share the same empty label $\lambda$. Figure 4.3(b) shows a Petri net constructed from the undirected graph in Fig. 4.3(a) using the above mentioned procedure. The construction of the Petri net from the given graph has complexity $O(|E| \times |Z|)$ that is polynomial in the number of edges and vertices. It is worth noting the following:

- The constructed partially observed Petri net $Q$ has $P_o = \{p_1, p_2, \ldots, p_{n_1}\}$, $T_o = \emptyset$, and labeling function $L(t_i) = \lambda$ for $i = 1, 2, \ldots, m$, which implies that all transitions are unobservable. Given this partially observed Petri net and a positive integer $k = l$ (note that $l$ is given in Problem 2.1), we have an instance of Problem 4.3. As there are no identically behaving transitions in the constructed Petri net $Q$ (because of
the way that the weights of arcs are assigned), we can decide whether a place sensor configuration $V$ is valid or not by checking transition distinguishability following Proposition 4.1.

- All transitions are labeled with the same empty label, and the input place and output place of each transition are observable (unobservable places are isolated). Each observable place can be used to distinguish all of its input and output transitions as the firing of these transitions will result in different token changes in the place (recall the procedure of assigning weights to arcs). Therefore, we can distinguish a transition by putting a sensor on either its input place or output place.

Note that there is a one-to-one correspondence between a valid place sensor configuration $V'$ for $Q$ and a vertex cover $Z'$ in $H$:

(i) Given any valid place sensor configuration $V'$ for $Q$, each transition must have at least one input or output place $p$ such that $V'(p) = 1$ (if $V'(p) = 0$ for both the input place and the output place of the transition, then the unobservable transition cannot be detected, and therefore the Petri net is not transition distinguishable); then, the set of vertices that correspond to the set of places that have sensors in $V'$ is also a vertex cover for the undirected graph $H$.

(ii) Given any vertex cover $Z'$, for each edge $e \in E$, at least one of its two vertices $v$ belongs to $Z'$; then, the transition corresponding to the edge $e$ in $H$ can be distinguished from other transitions by the place
corresponding to the selected vertex \( v \); therefore, we have established
the validity of place sensor configuration \( V' \) (with \( V'(p) = 1 \) iff \( p \)
corresponds to a vertex in the cover \( Z' \)).

This completes the proof of the \( NP \)-completeness.

\[ \square \]

**Example 4.4** For the undirected graph in Fig. 4.3(a), \( Z' = \{1, 2, 4, 6\} \) is
a vertex cover, and correspondingly, \( V' = (1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0)^T \) is a valid place
sensor configuration for the Petri net in Fig. 4.3(b).

\[ \square \]

4.5 Transformation of OPSS to Binary Integer Programming

In this section, we transform the OPSS problem to the binary integer pro-
gramming problem (for details, refer to Problem 2.2) so that it can be solved
optimally using existing binary integer programming solvers (this is possible
for small problem instances). Before we present the formal transformation,
we first use an example to illustrate the main idea.

**Example 4.5** For the Petri net in Fig. 4.1 with the OPSS problem intro-
duced in Example 4.2, we can formulate the following binary integer pro-
gramming problem corresponding to the OPSS problem:

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad Ax \geq b
\end{align*}
\]

where \( c = (1 \ 1 \ 1)^T \), \( x = V = (v_1 \ v_2 \ v_3)^T \) (as \( p_1 \), \( p_2 \) and \( p_3 \) are all observable),
\( b = (1 \ 1)^T \), and\(^4\)

\[
A = \begin{bmatrix}
-1 \neq -2 & 1 \neq 0 & 0 \neq 1 \\
0 \neq 0 & 0 \neq 0 & -1 \neq 0
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 1
\end{bmatrix}.
\]

The first row of matrix \( A \) is obtained by comparing the first three entries of
\( D(:, t_2) \) with the corresponding entries of \( D(:, t_3) \) and captures the require-
ment that the place sensor configuration \( V \) distinguish transitions \( t_2 \) and \( t_3 \);
the second row is obtained by comparing the first three entries of \( D(:, t_5) \) with

\(^4\)Here, for integers \( a \) and \( b \), \( a \neq b \) has value 1 if \( a \) is not equal to \( b \), and 0 otherwise.
three 0’s and captures the requirement that \( V \) detect unobservable transition \( t_5 \). Using the linear integer programming solver \([74]\), the optimal solution is found to be \((0 \ 0 \ 1)^T\), which is the same as the solution obtained via exhaustive search in Example 4.2.

To transform an instance of the OPSS problem to an instance of the BIP problem, we define the parameters of the BIP problem as follows:

- Set \( s = n_1 \) and \( q = |T_\lambda| + \sum_{e \in \Sigma \cup \{\lambda\},|T_e|\geq 2}(\frac{|T_e|}{2}) \).
- Set \( b = 1_q, c = 1_s \) and \( x = V \).
- Set \( A \) to be a \( q \times s \) binary matrix with two kinds of rows: (a) for each pair \( t_j, t_k \in T_e (j \neq k) \) for any label \( e \in \Sigma \cup \{\lambda\} \) with \(|T_e| \geq 2\), there is a row of the form \((D(1,j) \neq D(1,k) \ \ D(2,j) \neq D(2,k) \ \cdots \ D(n_1,j) \neq D(n_1,k))\); (b) for each \( t_j \in T_\lambda \), there is a row of the form \((D(1,j) \neq 0 \ \ D(2,j) \neq 0 \ \cdots \ D(n_1,j) \neq 0)\).

**Proposition 4.4** The binary integer programming problem constructed above is equivalent to the optimal place sensor selection problem.

**Proof:** As \( x = V \), \( c^T x = \sum_{i=1}^{n_1} v_i = ||V|| \). We need to show \( Ax \geq b \) if and only if \( V \) is valid, i.e., the Petri net is structurally observable under place sensor configuration \( V \) and the given labeling function \( L \). Following Proposition 4.1 and Proposition 4.2, we only need to show \( Ax \geq b \) if and only if all columns of \( D^e_\emptyset \) are pairwise different for each label \( e \in \Sigma \) such that \(|T_e| \geq 2\), and all columns of \( D^e_\lambda \) are nonzero and pairwise different. This is established from the following facts:

- \((D(1,j) \neq D(1,k) \ \ D(2,j) \neq D(2,k) \ \cdots \ D(n_1,j) \neq D(n_1,k))x \geq 1\) is equivalent to the fact that the \( j \)th and \( k \)th columns of incidence matrix \( D \) differ in at least one of the observable places that are equipped with a sensor (as indicated by the place sensor configuration \( V = x \)). The construction of matrix \( A \) ensures that this is true for all pairs \( t_j, t_k \in T_e \) (\( j \neq k \)) for any label \( e \in \Sigma \cup \{\lambda\} \) with \(|T_e| \geq 2\). In total, there are \( \sum_{e \in \Sigma \cup \{\lambda\},|T_e|\geq 2}(\frac{|T_e|}{2}) \) inequalities of this type.
- \((D(1,j) \neq 0 \ \ D(2,j) \neq 0 \ \cdots \ D(n_1,j) \neq 0)x \geq 1\) is equivalent to the fact that the \( j \)th column is nonzero in at least one of the observable
places that are equipped with a sensor. The construction of matrix $A$ ensures that this is true for any $t_j \in T_{\lambda}$. In total, there are $|T_{\lambda}|$ inequalities of this type.

Note that if we try to solve the equivalent binary integer programming problem without testing the existence condition to the OPSS problem, then the feasible region will be empty if there is no solution to the OPSS problem (and hopefully this will be indicated by the corresponding solver). It can be verified that the reduction from OPSS to BIP can be performed with complexity that is polynomial in the number of places and transitions. Therefore, we can solve the OPSS problem optimally using binary integer programming solvers. Though binary integer programming solvers will generally be more efficient than exhaustive search, they can only effectively deal with small problem instances as the OPSS problem is computationally hard. Therefore, it is imperative to find approximation algorithms that lead to good suboptimal solutions with reasonable computational effort. This is done in the next section.

4.6 Approximating OPSS: Part I

In this section, we propose three approximation algorithms for the optimal place sensor selection problem by extending the heuristics used for the optimal selection of diagnostic probes in [75]. More specifically, we consider how the addition of a sensor to an observable place serves to detect/distinguish unobservable transitions or distinguish transitions that are associated with the same label $e \in \Sigma$. We then define a measure of “distinguishability” (called scoring function) and use it to facilitate our choice of observable places to put sensors on. Note that due to the fact that structural observability is preserved with the addition of extra sensors, the algorithms we propose are guaranteed to find minimal sensor configurations (in the sense of [60]; for details, refer to Footnote 2 in Section 4.1).

Given a partially observed Petri net $Q$ with $n$ places, $m$ transitions, and a fixed labeling function $L$, if there exists one or more unobservable transitions, we construct a new partially observed Petri net $\tilde{Q}$ by adding an isolated transition $t_{m+1}$ to $Q$ and by assigning the empty label $\lambda$ to it. In other words, the new incidence matrix $\tilde{D} = (D \: 0_n)$ if there exists at least one unobservable
transition in $Q$; otherwise, $\bar{D} = D$. Given a place sensor configuration $V$, it can be shown (following Proposition 4.2) that if, for each label $e \in \Sigma \cup \{\lambda\}$, all columns of $\bar{D}^e_V$ are different from each other, then the original Petri net $Q$ is transition distinguishable. We introduce $\bar{D}$ because (i) it simplifies Proposition 4.2 in the sense that we only need to check if any two columns of $\bar{D}^e_V$ are different for $e \in \Sigma \cup \{\lambda\}$, including the empty label; and (ii) it eliminates the need to separately treat unobservable transitions (whose firings generate token changes at place $p$) from unobservable transitions (whose firings do not generate token changes at $p$) in the partition of $T_\lambda$ generated by observable place $p$ (as defined below).

**Definition 4.6** Given a partially observed Petri net $Q$ and a fixed labeling function $L$, $\Omega_e(p_i)$ for $e \in \Sigma \cup \{\lambda\}$ denotes the partition of $T_e$ generated by observable place $p_i$ in the Petri net $\bar{Q}$, and is defined as $\Omega_e(p_i) = \{S_1, S_2, ..., S_k\}$, where

- $k$ is equal to the number of distinct entries in the row vector $\bar{D}^e_V$, where $V$ has $V(i) = 1$ and all other entries equal to 0;
- $S_1 \cup S_2 \cup \cdots \cup S_k = T_e$ and $S_i \cap S_j = \emptyset$ for different $i, j$;
- $S_l$ for $l = 1, 2, ..., k$ is a non-empty set with the maximal number of transitions $\{t_j, ..., t_k\}$ that satisfy $t_j, \cdots, t_k \in T_e$ and $\bar{D}(i, j) = \cdots = \bar{D}(i, k)$.

**Example 4.6** We find the partitions generated by observable place $p_1$ in the Petri net constructed from the net in Fig. 4.1 by adding an isolated transition $t_6$. Then the row of the incidence matrix $\bar{D}$ corresponding to place $p_1$ is

\[
\begin{bmatrix}
1 & -1 & -2 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{bmatrix},
\]

and, by definition, the partitions generated by place $p_1$ for labels $a, b, c$ and $\lambda$ are respectively $\{\{t_1\}\}$, $\{\{t_2\}\}$, $\{\{t_3\}\}$, $\{\{t_4\}\}$ and $\{\{t_5, t_6\}\}$.

**Remark 4.4** The partition defined above generalizes the measure of the diagnostic power of a probe in [75] in the sense that the incidence matrix $\bar{D}$ can have arbitrary integer entries while the dependency matrix in [75] only has 0 or 1 entries.
If there exists at least one unobservable transition in $Q$, then $|T_{\lambda}| \geq 2$ in $\bar{Q}$. Clearly for a label $e$ with only one transition, the transition is observable and uniquely identified by its label. The transitions associated with a label $e \in \Sigma \cup \{\lambda\}$ satisfying $|T_e| \geq 2$ need to be distinguished using place sensors so that transition distinguishability holds. If the partition generated by an observable place $p$ for such a label $e$ has $|T_e|$ elements, then a sensor at place $p$ is sufficient for distinguishing all transitions in $T_e$. In Example 4.6, $p_1$ is sufficient for distinguishing transitions associated with label $b$, but is insufficient for distinguishing transitions associated with $\lambda$.

Now we define the scoring function for an observable place $p_i$ as

$$f(p_i) = \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} |\Omega_e(p_i)|.$$ 

The scoring function satisfies

$$l_b \leq f(p_i) \leq u_b,$$

where $l_b = \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} 1$ and $u_b = \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} |T_e|$, because $1 \leq |\Omega_e(p_i)| \leq |T_e|$. If $f(p_i) > f(p_j)$, then a place sensor at $p_i$ can distinguish more (partitions of) transitions than a place sensor at $p_j$; therefore, $f(p_i)$ is a measure of the ability of place $p_i$ to distinguish transitions. If $f(p_i)$ is equal to $u_b$, then $p_i$ is sufficient for distinguishing all transitions.

With the scoring functions computed, one natural top-down method to approximate the OPSS problem is to start with all observable places and then subtract places one by one in the order of increasing value of the scoring function, until we reach a set of observable places (to place sensors on) that cannot be reduced further without affecting the transition distinguishability of the net. The details are given in Algorithm 5.

We briefly explain Lines 4-12 of Algorithm 5. After computing the score for every observable place, we set $V_{\text{current}}$ to be $V_{\text{max}}$ and $P_{\text{left}}$ to be the set of observable places at Line 4. $P_{\text{left}}$ keeps track of the set of observable places that we have not considered so far and its cardinality decreases by one after each loop in the while clause; this guarantees the termination of the algorithm. In the loop, we use $V_{\text{temp}}$ to represent the place sensor

---

5Note that other scoring functions are also possible; we discuss some possible choices in Chapter 5.2.
Algorithm 5  Top-down method

**Input:** A partially observed Petri net $Q$ and a fixed labeling function $L$

**Output:** $V$ – an approximation of $V_{\min}$

1: Determine whether there exists an optimal place sensor configuration using Theorem 4.1; if one does not exist, exit;
2: Construct $\overline{Q}$ by adding an isolated transition if there is at least one unobservable transition;
3: Compute $f(p_i)$ for every $p_i \in P_o$;
4: $V_{\text{current}} \leftarrow V_{\text{max}}$ and $P_{\text{left}} \leftarrow P_o$;
5: while $|P_{\text{left}}| > 0$ do
6: $V_{\text{temp}} \leftarrow V_{\text{current}}$;
7: Find $p \in P_{\text{left}}$ to minimize $f(p)$. If there are multiple places that minimize $f(p)$, randomly choose one;
8: $V_{\text{temp}}(p) \leftarrow 0$;
9: Test whether $\overline{Q}$ is transition distinguishable under $V_{\text{temp}}$. If it is, then $V_{\text{current}} \leftarrow V_{\text{temp}}$;
10: $P_{\text{left}} \leftarrow P_{\text{left}} - \{p\}$;
11: end while
12: $V \leftarrow V_{\text{current}}$.

configuration induced from $V_{\text{current}}$ by removing a sensor from the place $p$ that minimizes the score among all places in $P_{\text{left}}$. If the system is transition distinguishable under $V_{\text{temp}}$ and $L$, then the sensor can be removed and we update $V_{\text{current}}$ using $V_{\text{temp}}$; otherwise, the sensor cannot be removed at the current iteration and $V_{\text{current}}$ remains the same as before. In both cases, we remove the corresponding place $p$ from $P_{\text{left}}$. After we have considered all observable places, i.e., $P_{\text{left}}$ is empty, the algorithm stops with the output $V_{\text{current}}$, which is an approximation to $V_{\min}$ because in any execution of the **while** loop, $V_{\text{current}}$ is always valid.

For certain Petri nets, it may be the case that some $f(p_i)$ is very close to $u_b$ so that one or two additional place sensors are sufficient to guarantee structural observability. We next consider how to extend the scoring function to multiple places so that we can approximate the optimal solution in a bottom-up fashion. Suppose we have a partition $\Omega_e(p_i)$ generated by place $p_i$ for label $e$ and $|\Omega_e(p_i)| < |T_e|$. If we choose another place $p_j$, we can refine

\[\text{6If } |\Omega_e(p_i)| = |T_e| \text{ for every label } e, \text{ then adding another place cannot increase the value of the scoring function. In fact, in such a case, a sensor at place } p_i \text{ is sufficient for structural observability and the proposed Algorithm 6 would terminate after selecting } p_i.\]
Algorithm 6 Bottom-up method

**Input:** A partially observed Petri net \( Q \) and a fixed labeling function \( L \)

**Output:** \( V - \) an approximation of \( V_{\text{min}} \)

1: Determine whether there exists an optimal place sensor configuration using Theorem 4.1; if one does not exist, exit;
2: Construct \( \bar{Q} \) by adding an isolated transition if there is at least one unobservable transition;
3: \( V_{\text{current}} \leftarrow 0 \). If \( \bar{Q} \) is transition distinguishable under \( L \) and \( V_{\text{current}} \), exit with \( V = 0 \); 
4: \( P_{\text{left}} \leftarrow P_{\text{o}}, S \leftarrow 0 \), and sign \( \leftarrow 0 \);
5: **while** sign is 0 **do**
6: Find \( p \in P_{\text{left}} \) to maximize \( f(S \cup \{p\}) \) and keep their generated partitions for each label. If there are multiple places that result in an equal maximum, randomly choose one;
7: \( V_{\text{current}}(p) \leftarrow 1 \), \( P_{\text{left}} \leftarrow P_{\text{left}} - \{p\} \) and \( S \leftarrow S \cup \{p\} \);
8: If \( f(S) \) is equal to \( u_b \), then sign \( \leftarrow 1 \);
9: **end while**
10: \( V \leftarrow V_{\text{current}} \).

the partition \( \Omega_e(p_i) \) by considering whether transitions belonging to some set \( S \in \Omega_e(p_i) \) can be distinguished using \( p_j \). If we denote the refined partition as \( \Omega_e(p_i, p_j) \), we can define \( f(\{p_i, p_j\}) = \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} |\Omega_e(p_i, p_j)| \). By repeatedly applying the above operation, we can define the scoring function \( f(S) \) for any nonempty set \( S \) of observable places. Note that the partitions in \( \Omega_e(p_{i_1}, p_{i_2}, \ldots, p_{i_k}) \) are independent of the ordering of places and \( f(S) \) still satisfies \( l_b \leq f(S) \leq u_b \) for any nonempty set of observable places.

The idea of the bottom-up method is as follows: initially, choose the place \( p_i \) which maximizes \( f(p_i) \) and keep its generated partition for each label; the second time around, choose the place \( p_j \) which maximizes \( f(\{p_i, p_j\}) \) and keep the refined partition for each label; keep doing this until \( f(S) = u_b \) for some subset \( S \) of observable places. The details are given in Algorithm 6.

We briefly explain Lines 3-10 of Algorithm 6. We first set \( V_{\text{current}} \) to be the place sensor configuration without sensors at Line 3. If \( \bar{Q} \) is transition distinguishable under the given labeling function \( L \) and the place sensor configuration \( V_{\text{current}} \), then no place sensor is necessary and the algorithm exits with \( V = 0 \); otherwise we set \( P_{\text{left}} \) to be the set of observable places, \( S \) to be the empty set, and sign to be 0 at Line 4. \( P_{\text{left}} \) keeps track of
the set of observable places that we have not considered so far, variable $S$ keeps track of the set of observable places that have sensors in $V_{current}$, and $sign$ has value 1 if we have found a valid place sensor configuration and 0 otherwise. In the while loop, we first select place $p$ (in $P_{left}$) that maximizes $f(S \cup \{p\})$; then, we set $V_{current}(p)$ to be 1, remove $p$ from $P_{left}$, and at the same time add $p$ into $S$; finally, we determine if $f(S)$ is equal to $u_b$, or equivalently, if $V_{current}$ is valid. If it is, then the while loop ends by setting $sign$ to be 1 and declaring that $V_{current}$ is an approximation to $V_{min}$; otherwise, the algorithm goes to the next loop. The algorithm is guaranteed to stop because after each iteration, $|S|$ will be increased by 1 and $|S|$ is upper bounded by the number of observable places (and also because the existence of an optimal solution has been verified at Line 1).

Example 4.7 For the Petri net in Fig. 4.1 with the OPSS problem introduced in Example 4.2, we illustrate the use of the top-down and the bottom-up methods to solve the OPSS problem. The existence of an optimal solution was shown in Example 4.2. As $t_5$ is unobservable, we add an isolated transition $t_6$ to construct $\bar{Q}$.

We first illustrate the use of the top-down method. We compute $f(p_i)$ for $i = 1, 2, 3$, and obtain $f(p_1) = 3$, $f(p_2) = 3$ and $f(p_3) = 4$. At the first iteration, $p_1$ and $p_2$ both minimize $f(p)$; we choose to eliminate the sensor on $p_1$ and find out that $\bar{Q}$ is still transition distinguishable. At the second iteration, we eliminate the sensor on $p_2$ while at the third iteration, we cannot eliminate any sensor and the algorithm ends with $V = (0 \ 0 \ 1)^T$.

To use the bottom-up method, we initialize $S = \emptyset$. At the first iteration, we find $p_3$ which maximizes $f(p)$ and set $S = \{p_3\}$. As $f(S) = u_b = 4$, the algorithm ends with the $V = (0 \ 0 \ 1)^T$. For this particular example, both algorithms give the optimal solution that we obtained earlier using exhaustive search (in Example 4.2) and binary integer programming solvers (in Example 4.5).

Now we provide an analysis of the performance of the bottom-up method by providing an upper bound on the number of place sensors in the solution generated by Algorithm 6. The upper bound is $\min(n_1, u_b - l_b + 1)$, where

\footnote{Instead of computing all partitions generated by $S \cup \{p\}$, we can save computation by computing the value of $f(S \cup \{p\})$ based on the value of $f(S)$ and the increment induced by place $p$ (recall that Algorithm 6 iterates through places in a bottom-up fashion).}
$n_1$ is the number of observable places and $u_b$ (or $l_b$) is the maximum (or minimum) value of the scoring function. The reasoning is the following: at each iteration in Algorithm 6, one place sensor will be added and the scoring function increases by at least $l_b$ for the first picked place and by at least 1 for the following picked places (otherwise, we will not select the place sensor); therefore, the loop will be executed at most $n_1$ times as there can be at most $n_1$ sensors to add, and also at most $u_b - l_b + 1$ times as the value of the scoring function increases from at least $l_b$ to $u_b$ with the increment being at least 1.

For the OPSS problem in Example 4.7, as $n_1 = 3$, $u_b = 4$ and $l_b = 2$, we get the upper bound $\min(n_1, u_b - l_b + 1) = \min(3, 4 - 2 + 1) = 3$; clearly, the bound holds as there is only one sensor in the optimal solution.

As shown in Appendix C, both Algorithm 5 and Algorithm 6 have complexity $O(n^2m^2)$ that is polynomial in the number of places and transitions. We can obtain a better solution if we apply Algorithm 5 after Algorithm 6 by setting $V_{current}$ (at Line 4 of Algorithm 5) to be the output of Algorithm 6 (instead of $V_{max}$); this method is called the combined method and is illustrated in the context of an application in Chapter 5.2.

4.7 Approximating OPSS: Part II

In the previous section, we introduced the top-down, the bottom-up, and the combined methods, but we have not been able to establish the performance guarantees for these three approximation algorithms. In this section, we propose a different heuristic algorithm based on the reduction from the OPSS problem to the set cover problem (for details, refer to Problem 2.3) and the use of a well known greedy algorithm for the set cover problem [24]. This results in yet another heuristic method that has the advantage of providing performance guarantees.

We first use the following example to illustrate the basic idea of how to convert an instance of the OPSS problem to an instance of the set cover problem.

**Example 4.8** Consider the partially observed Petri net $Q$ in Fig. 4.1 with the OPSS problem introduced in Example 4.2. The OPSS problem requires a place sensor configuration $V$ such that $Q$ is structurally observable under
Given an instance of the OPSS problem, one could construct an instance of the set cover problem as shown in Algorithm 7. Now we briefly show the correctness of the reduction. Note that for any place sensor configuration $V$, there is a unique combination of subsets of $U$ (namely $\{S_i \mid V(i) = 1\}$), and vice versa. Also, $Q$ being structurally observable under $V$ and $L$ is equivalent to satisfying $AV \geq 1_q$, which is equivalent to the requirement that the combination of subsets of $U$ (corresponding to $V$) covers every element of the universe $U$. Most computation in the reduction comes from Step 3 which involves constructing the matrix $A$ and can be verified to have com-
Algorithm 7 Reduction from OPSS to SCP

Input: An instance of the OPSS problem as shown in Problem 4.1
Output: An instance of the set cover problem as shown in Problem 2.3

1: Set \( q = |T_\lambda| + \sum_{e \in \Sigma \cup \{\lambda\}, |T_e| \geq 2} \binom{|T_e|}{2} \), and set \( k = n_1 \) (namely, the number of observable places);
2: Set \( U = \{1, 2, ..., q\} \);
3: Set \( A \) to be a \( q \times n_1 \) binary matrix with two kinds of rows: a) for each pair \( t_j, t_k \in T_e \) \((j \neq k)\) for any label \( e \in \Sigma \cup \{\lambda\} \) with \(|T_e| \geq 2\), there is a row of the form \((D(1,j) \neq D(1,k) D(2,j) \neq D(2,k) \cdots D(n_1,j) \neq D(n_1,k))\);
   b) for each \( t_j \in T_\lambda \), there is a row of the form \((D(1,j) \neq 0 D(2,j) \neq 0 \cdots D(n_1,j) \neq 0)\);
4: Set \( S_i = \{j \mid j \in U \text{ and } A(j,i) = 1\} \) for \( i = 1, 2, ..., n_1 \);
5: Output the set cover problem.

Complexity \( O(nm^2) \) because \( q \) is \( O(m^2) \). Clearly, the reduction is of polynomial complexity.

Remark 4.5 Note that one can also reduce the set cover problem to the OPSS problem (this is another way to establish the NP-completeness of the OPSS problem instead of using the vertex cover problem as shown in Section 4.4). Here is a sketch of the reduction: given a universe \( U = \{1, 2, ..., q\} \) and a set of subsets \( S_1, S_2, ..., S_k \), we construct a partially observed Petri net \( Q \) such that the partially observed Petri net has

- 2q observable transitions,
- q labels (namely, \(|\Sigma| = q\)) and label \( e_i \) covering a unique pair of transitions \( t_{2i-1} \) and \( t_{2i} \) for \( i = 1, ..., q \),
- k observable places, and
- \( D(j, 2i-1) \neq D(j, 2i) \) if the subset \( S_j \) covers the element \( i \) in the universe \( U \) and \( D(j, 2i-1) = D(j, 2i) \) otherwise, for \( j = 1, ..., k \) and \( i = 1, ..., q \).

One unobservable place might be necessary to eliminate identically behaving transitions in the above construction. The correctness can be verified based on the discussion of the reduction from the OPSS problem to the set cover problem. ■
There is a well-known greedy algorithm for the set cover problem which selects at each time the subset $S_i$ that can cover the most elements in the universe that have not been covered so far, and terminates when all elements are covered. The algorithm is guaranteed to provide a solution within $OPT^* H_q$ [24], where $OPT$ is the minimum number of subsets and $H_q = 1 + \frac{1}{2} + \ldots + \frac{1}{q}$ (note that $H_q$ is $O(\ln q)$).

When we first reduce the OPSS problem to the set cover problem, and then utilize the known greedy algorithm, the method guarantees a place sensor configuration with the number of place sensors within $OPT^* H_q$, where $OPT$ is the minimum number of place sensors and $q = |T_\lambda| + \sum_{\epsilon \in \Sigma \cup \{\lambda\}, |T_\epsilon| \geq 2} (\binom{|T_\epsilon|}{2})$ as shown in Algorithm 7. Note that the factor $H_q$ is roughly $O(\ln m)$ because $q$ is $O(m^2)$.

4.8 Optimal Transition Sensor Selection

Section 4.4 showed that the optimal place sensor selection problem is computationally hard by showing that the corresponding decision problem is $\mathcal{NP}$-complete. Perhaps surprisingly, the optimal transition sensor selection problem (as defined in Problem 4.2) is solvable with complexity that is polynomial in the number of places and transitions. We first define the partition of $T_o$ generated by a place sensor configuration $V$, which is a slightly modified version of Definition 4.6.

**Definition 4.7** Given a partially observed Petri net $Q$ and a fixed place sensor configuration $V$, the partition of the set of observable transitions $T_o$ generated by $V$ is defined to be $\Omega(V) = \{S_0, S_1, S_2, \ldots, S_k\}$, where

- $S_0$ is a (possibly empty) set with the maximal number of transitions $\{t_j, \ldots, t_l\}$ that satisfy $t_j, \ldots, t_l \in T_o$ and $D_V(:, j) = \cdots = D_V(:, l) = 0_{||V||}$ (where $D_V(:, j)$ denotes the $j$th column of matrix $D_V$);
- If $S_0 = \emptyset$, then $k$ is equal to the number of distinct columns in the matrix $D_V$; if $S_0 = T_o$, then $k$ is defined to be 1 and $S_1 := \emptyset$; otherwise, $k$ is equal to the number of distinct columns of matrix $D_V$ minus 1;
- $S_0 \cup S_1 \cup S_2 \cup \cdots \cup S_k = T_o$ and $S_i \cap S_j = \emptyset$ if $i \neq j$.
\begin{itemize}
  \item \(S_i\) for \(i = 1, 2, \ldots, k\) is a (possibly empty) set with the maximal number of transitions \(\{t_j, \ldots, t_l\}\) that satisfy \(t_j, \cdots, t_l \in T_o\) and \(D_V(:, j) = \cdots = D_V(:, l)\) being nonzero.
\end{itemize}

Essentially, any two transitions that have the same column in \(D_V\) fall into the same partition in \(\Omega(V)\). After computing the partition \(\Omega(V)\), we need to configure transition sensors to distinguish/detect all transitions in \(S_0\) by assigning a unique label to each transition in \(S_0\) because no token changes are available for these transitions; we also need to configure transition sensors to distinguish all transitions in \(S_i\) for \(i = 1, 2, \ldots, k\) by assigning a unique label to any \(|S_i| - 1\) transitions in \(S_i\) (the transition that is left without a label can be distinguished by the token changes in \(V\)). The minimum number of labels needed is equal to

\[
|\Sigma|_{\text{min}} = \max(|S_0|, \max(|S_1|, |S_2|, \ldots, |S_k|) - 1).
\]

If \(|\Sigma| < |\Sigma|_{\text{min}}\) (i.e., if the number of available transition sensors is less than \(|\Sigma|_{\text{min}}\)), then by Dirichlet’s drawer principle, either some transition in \(S_0\) cannot be detected or at least two transitions in some \(S_l\) for \(1 \leq l \leq k\) cannot be distinguished. Based on this idea, we have Algorithm 8 for the OTSS problem.

We briefly explain Lines 3-12 of Algorithm 8. After computing \(\Omega(V)\), at Line 3 we set \(|\Sigma|\) (the number of transition sensors) to be

\[
\max(|S_0|, \max(|S_1|, |S_2|, \ldots, |S_k|) - 1)
\]

as discussed previously. If \(|\Sigma| = 0\), then no transition sensor is required; otherwise, we need \(|\Sigma|\) transition labels \(e_1, e_2, \ldots, e_{|\Sigma|}\). Then we define the labeling function \(L_{\text{min}}\) for observable transitions in Lines 4-11 (and for unobservable transitions at Line 12). At Line 4 we assign a unique label to each transition in \(S_0\) if \(S_0\) is not empty. In the \textbf{for} loop from Line 5 to Line 11, we assign labels to each transition in \(S_l\) for \(l = 1, \ldots, k\). There are two cases:

(i) If \(|S_l| = 1\), then the token change can uniquely identify the only transition in \(S_l\), and therefore we assign the empty label to this transition.

(ii) If \(|S_l| > 1\), then we assign a unique label from \(|\Sigma|\) to each transition.
Algorithm 8 Algorithm for OTSS

**Input:** A partially observed Petri net $Q$ and a fixed place sensor configuration $V$

**Output:** A valid labeling function $L_{min} : T \rightarrow \Sigma \cup \{\lambda\}$ satisfying $L_{min}(t) = \lambda$ for any $t \in T_{uo}$

1. Determine whether there exists an optimal labeling function using Theorem 4.2; if one does not exist, exit;
2. Compute the partition $\Omega(V)$ to get $S_0, S_1, S_2, \ldots, S_k$;
3. $|\Sigma| \Leftarrow \max(|S_0|, \max(|S_1|, |S_2|, \ldots, |S_k|) - 1)$. If $|\Sigma| = 0$, exit with $L_{min}(t) = \lambda$ for $t \in T$; else, $\Sigma \Leftarrow \{e_1, e_2, \ldots, e_{|\Sigma|}\}$;
4. Assign a unique label from $\Sigma$ to each transition in $S_0$ if $S_0$ is nonempty;
5. for $l = 1$ to $k$ do
6. if $|S_l| = 1$ then
7. Assign the empty label to the unique transition in $S_l$;
8. else if $|S_l| > 1$ then
9. Assign a unique label from $\Sigma$ to each transition among any $|S_l| - 1$ transitions in $S_l$, and assign the empty label to the remaining transition;
10. end if
11. end for
12. $L_{min}(t) \Leftarrow \lambda$ for $t \in T_{uo}$ and output $L_{min}$.

The way to assign transition labels is possible due to the value of $|\Sigma|$. $L_{min}$ and the given $V$ guarantee transition distinguishability, and therefore $L_{min}$ is valid (and the number of transition labels is also minimized). Note that $L_{min}$ is not necessarily unique as there are no constraints on how to assign labels for transitions within a set $S_l$ or two different sets $S_{l_1}$ and $S_{l_2}$.

**Example 4.9** We consider the OTSS problem as stated in Example 4.3. Recall that the optimal labeling function exists if we are given place sensor configuration $V = (0 \ 0 \ 1)^T$. The partition of $T_o$ generated by $V$ is $S_0 = \{t_1, t_2\}$ and $S_1 = \{t_3, t_4\}$ by examining the matrix $D_V = (0 \ 0 \ 1 \ 1 \ -1)$. Therefore, we can set $|\Sigma| = \max(|S_0|, |S_1| - 1) = 2$ and $\Sigma = \{e_1, e_2\}$. Labeling function $L$ defined as $L(t_1) = L(t_3) = e_1$, $L(t_2) = e_2$ and $L(t_4) = L(t_5) = \lambda$ is one optimal solution. □
As shown in Appendix D, this algorithm has complexity $O(nm^2)$ that is polynomial in the number of places and transitions.

**Remark 4.6** Though there can be multiple optimal labeling functions, we may form preferences depending on other criteria; for example, using the results in Chapter 3, we may want to minimize the number of possible states that are consistent with the observation of labels in case of place sensor failures.

In the OTSS problem, we have made one implicit assumption: a nonempty label can be associated to any subset of observable transitions. This assumption may not be realistic in certain applications due to other constraints (one constrained OTSS problem is studied in the next section). For example, the labeling function may be required to be injective over observable transitions, which means every observable transition should be associated with a unique (possibly empty) label. For the OTSS problem under this constraint, it can be shown (following almost the same reasoning as for the OTSS problem) that the minimum number of transition labels $|\Sigma|$ is $|S_0| + \max(|S_1| + |S_2| + \cdots + |S_k| - 1, 0)$ as we can assign the empty label to one of these observable transitions (which cause token changes in observable places with sensors). More specifically, $|\Sigma| = |T_o|$ if $S_0 = T_o$, and $|\Sigma| = |T_o| - 1$ otherwise.

### 4.9 General Sensor Selection

In this section, we look at the general sensor selection problem mentioned in Section 4.2. Our goal is to choose a set of place sensors and transition sensors of minimum cardinality such that the system state can be determined uniquely at any time step based on sensor information (and knowledge of the system model and initial state). For this general case, using Theorem 4.1 and Theorem 4.2, it is easy to verify the following condition for the existence of an optimal solution.

**Corollary 4.1** Given a partially observed Petri net $Q$, there exists an optimal sensor selection for the general sensor selection problem if and only if $Q$ is transition distinguishable under the place sensor configuration $V_{max}$ and the labeling function $L_{max}$.
Based on the result in Section 4.4, the general problem is also computationally hard. The problem can be solved optimally by exhaustively searching all possibilities. More specifically, one can first choose a place sensor configuration \( V \) (among \( 2^n \) possible choices), solve the OTSS problem with the fixed place sensor configuration \( V \), obtain the minimum number of transition labels required, and finally add the number of place sensors and the number of transition labels, and select the minimum sum among all \( 2^n \) choices of place sensor configurations. However, it is not quite clear how to transform the problem into an integer programming problem due to the heterogeneity of place sensor configurations and transition labeling functions.

Interestingly, if we have certain constraints on the transition sensors, we can show that constrained OTSS problem and constrained general sensor selection problem (both will be defined shortly) can be converted to an OPSS problem.

In the OTSS problem, we have one implicit assumption: a nonempty label can be associated to any subset of observable transitions. This assumption may not be realistic in certain applications due to topological, or other constraints; for instance, it might be the case that only physically close transitions can share the same label. Therefore, we now consider how the problem changes if the following constraints are imposed on transition sensors:

(i) There are \( d \) types of transition sensors \( T_1, T_2, ..., T_d \).

(ii) Each type \( T_i \) covers a subset of observable transitions while some transitions may not be covered and some transitions may be covered by more than one type of sensor.

If a transition \( t \) is covered by a type \( T_i \) transition sensor, then the label \( e_{T_i} \) will be observed if \( t \) fires; if \( t \) is covered by more than one type of transition sensor (e.g., covered by both type \( T_i \) and type \( T_j \) transition sensors), then all associated labels will be simultaneously observed if \( t \) fires (e.g., labels \( e_{T_i} \) and \( e_{T_j} \) will be observed simultaneously, or equivalently, a single label \( e_{T_i T_j} \) will be observed).

The transition sensor configuration \( W \) is a vector \((w_1, w_2, ..., w_d)^T\), where \( w_i = 0 \) if no type \( i \) transition sensor exists for transitions in \( T_i \) and \( W_i = 1 \) otherwise. \( ||W|| := \sum_{i=1}^d w_i \leq d \) denotes the total number of transition sensors in the transition sensor configuration \( W \). Given a transition sensor
configuration $W$, we can construct an equivalent labeling function $L_W$ as shown in the following example.

**Example 4.10** For the partially observed Petri net on the left of Fig. 2.3, suppose there are two types of transition sensors: $T_1$ (which covers transitions $t_1$ and $t_2$) and $T_2$ (which covers transitions $t_2$ and $t_3$). If $W = (11)^T$, then the equivalent labeling function $L_W$ is $L_W(t_1) = e_{T_1}$, $L_W(t_2) = e_{T_1T_2}$, $L_W(t_3) = e_{T_2}$, $L_W(t_4) = L_W(t_5) = \lambda$. The labeling function is equivalent to the transition sensor configuration in the sense that the outputs from both are essentially the same given the same system activities. It is straightforward to generalize the construction of $L_W$ to an arbitrary transition sensor configuration $W$.

Now we are ready to formulate the constrained optimal transition sensor selection problem.

**Problem 4.4** (Constrained Optimal Transition Sensor Selection (COTSS)) Given a partially observed Petri net $Q$, a fixed place sensor configuration $V$ and $d$ types of transition sensors $T_1, T_2, ..., T_d$, find $W_{\text{min}}$ such that (i) the system is structurally observable under $V$ and $L_{W_{\text{min}}}$, and (ii) $W_{\text{min}}$ minimizes the number of transition sensors $||W_{\text{min}}||$.

More generally, we could have the following constrained general sensor selection problem.

**Problem 4.5** (Constrained General Sensor Selection (CGSS)) Given a partially observed Petri net $Q$, and $d$ types of transition sensors $T_1, T_2, ..., T_d$, find $V$ and $W$ such that (i) the system is structurally observable under $V$ and $L_W$, and (ii) $V$ and $W$ minimize the total number of sensors (namely, $||V|| + ||W||$).

Now we show that both the constrained optimal transition sensor selection problem and the constrained general sensor selection problem can be converted to the optimal place sensor selection problem.

### 4.9.1 Constrained Optimal Transition Sensor Selection

In the constrained OTSS problem, we have a fixed place sensor configuration $V$ which allows certain transitions to be distinguished. One can thus
construct a labeling function $L_V$ which provides essentially the same sensor outputs as the place sensor configuration $V$. Before we introduce the construction, we first look at the partition of $T$ generated by $V$, which is a slightly modified version of Definition 4.7.

**Definition 4.8** Given a partially observed Petri net $Q$ and a fixed place sensor configuration $V$, the *partition of $T$ generated by $V$* is defined to be $\Omega(V) = \{S_0, S_1, S_2, ..., S_k\}$, where

- $S_0$ is a (possibly empty) set with the maximal number of transitions $\{t_j, ..., t_l\}$ that satisfy $t_j, ..., t_l \in T$ and $D_V(:, j) = \cdots = D_V(:, l) = 0_{|V|}$;
- If $S_0 = \emptyset$, then $k$ is equal to the number of distinct columns in the matrix $D_V$; otherwise, $k$ is equal to the number of distinct columns of matrix $D_V$ minus 1;
- $S_0 \cup S_1 \cup S_2 \cup \cdots \cup S_k = T$ and $S_i \cap S_j = \emptyset$ if $i \neq j$;
- $S_i$ for $i = 1, 2, ..., k$ is a set with the maximal number of transitions $\{t_j, ..., t_l\}$ that satisfy $t_j, ..., t_l \in T$ and $D_V(:, j) = \cdots = D_V(:, l)$ being nonzero.

The firings of transitions in $S_0$ cannot generate any place sensor output; equivalently, we could assign the empty label to all transitions in $S_0$, i.e., $L_V(t) = \lambda$ for $t \in S_0$. For each $S_i$, $i = 1, 2, ..., k$, the firings of transitions in $S_i$ generate a unique combination of token changes among all places with sensors in $V$ (but this combination can be generated by any of these transitions); equivalently, we could assign the label $e_{S_i}$ to all transitions in $S_i$, i.e., $L_V(t) = e_{S_i}$ for $t \in S_i$.

Once we have the equivalent labeling function, we could construct an instance of the OPSS problem given a COTSS instance, as illustrated in the following example.

**Example 4.11** Consider the partially observed Petri net $Q$ on the left of Fig. 2.3 with a fixed place sensor configuration $V = (0 \ 0 \ 1)^T$, and 2 types of transition sensors with $T_1 = \{t_1, t_2\}$ and $T_2 = \{t_2, t_3\}$. The partition of $T$ generated by $V$ is $S_0 = \{t_1, t_2\}$, $S_1 = \{t_3, t_4\}$ and $S_2 = \{t_5\}$, as can be readily obtained from matrix $D_V = (0 \ 0 \ 1 \ 1 \ -1)$. The equivalent labeling function
$L_V$ is $L_V(t_1) = L_V(t_2) = \lambda$, $L_V(t_3) = L_V(t_4) = e_{s_1}$, and $L_V(t_5) = e_{s_2}$. The COTSS problem can be interpreted as follows: Given a labeling function $L_V$, find a transition sensor configuration $W$ such that all transitions are distinguished and $|W|$ is minimized.

Now we construct an instance of the OPSS problem from this COTSS problem. Consider a partially observed Petri net $Q'$ with 5 observable transitions $t'_1, ..., t'_5$ corresponding to $t_1, ..., t_5$, 2 observable places $p'_1, p'_2$ corresponding to the two types of transition sensors, and labeling function defined as $L'(t'_1) = L'(t'_2) = \lambda$, $L'(t'_3) = L'(t'_4) = e_{s_1}$, $L'(t'_5) = e_{s_2}$, which is essentially $L_V$. The incidence matrix $D'$ is

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{bmatrix}
\]

and is constructed based on the coverage of different types of transition sensors; for example, as sensor type $T_1$ can monitor $t_1, t_2$, the place $p'_1$ corresponding to $T_1$ can distinguish $t'_1, t'_2$ from other transitions and $D'(p'_1,:) = (1 1 0 0 0)$.

Note that transitions $t'_4$ and $t'_5$ are identically behaving transitions. One could add one unobservable place $p'_3$ with $D'(p'_3,:) = (0 0 1 -1)$ to resolve this issue.

Now the goal in the constructed OPSS problem is to find a $V'_{min}$ such that all transitions can be distinguished. For this example, it is easy to see that $Q'$ being structurally observable under $V'$ and $L_W$ is equivalent to $Q$ being structurally observable under $V := W$ and $L'$.

In general, given an instance of the COTSS problem, one can construct an instance of the OPSS problem as shown in Algorithm 9. Now we briefly argue the correctness of the reduction. Note that for any transition sensor

---

8The constraint we have regarding $D'(p'_1,:)$ is that the first two entries should be the same and nonzero, and the last three entries should all be zero. Therefore, other choices could also be possible (e.g., $D'(p'_1,:) = (-1 -1 0 0 0)$).

9The constraint we have regarding $D'(p'_3,:)$ is that $D'(p'_3,t'_4)$ and $D'(p'_3,t'_5)$ should be different.
Algorithm 9 Reduction from COTSS to OPSS

Input: An instance of the COTSS problem as shown in Problem 4.4
Output: An instance of the OPSS problem as shown in Problem 4.1

1: Calculate $\Omega(V) = \{S_0, S_1, ..., S_k\}$ based on Definition 4.8, and construct the labeling function $L_V$ satisfying $L_V(t) = \lambda$ if $t \in S_0$ and $L_V(t) = e_{S_i}$ if $t \in S_i$.
2: Construct $Q'': T''_o := \{t'_1, t'_2, ..., t'_m\}$, $P''_o := \{p''_1, p''_2, ..., p''_d\}$, $L''(t'_i) = L_V(t_i)$, and $D''(p''_i, t'_j) = 1$ (for $i = 1, ..., d$ and $j = 1, ..., m$) if $t_j$ is covered by $T_i$ and $D''(p''_i, t'_j) = 0$ otherwise.
3: Check if there are identically behaving transitions. If true, then add unobservable place $p_{d+1}'$ (i.e., $P' = P''_o \cup \{p_{d+1}'\}$) and assign $D''(p_{d+1}', ::)$ so that identically behaving transitions are eliminated; otherwise, $P' = P''_o$.
4: Output the OPSS problem instance with partially observed Petri net $Q'$ and fixed labeling function $L'$.

Configuration $W$ in $Q$, there is a place sensor configuration $V' := W$ in $Q'$ because observable place $p'_i$ in $Q'$ corresponds to the type of transition sensor $T_i$ in $Q$, and vice versa. Also, $Q$ being structurally observable under $V$ and $L_W$ is equivalent to $Q'$ being structurally observable under $L'$ and $V' := W$, because the construction of $L'$ provides the same distinguishability on transitions as the given $V$ (also note that $L'$ is constructed from $L_V$ and $L_V$ is equivalent to $V$). Most computation in the reduction comes from Step 1 and Step 2: Step 1 involves analyzing the matrix $D$ and can be verified to have complexity $O(nm^2)$; Step 2 involves constructing the new partially observed Petri net $Q'$ and can be verified to have complexity $O(dm)$, where $d$ is the number of types of transition sensors and $m$ is the number of transitions. Therefore, the reduction is of polynomial complexity.

4.9.2 Constrained General Sensor Selection

We use the following example to illustrate the basic idea of how to convert a CGSS problem instance to an OPSS problem instance.

Example 4.12 Consider the partially observed Petri net $Q$ on the left of Fig. 2.3 with 2 types of transition sensors satisfying $T_1 = \{t_1, t_2\}$ and $T_2 = \{t_2, t_3\}$. The CGSS problem asks to obtain a place sensor configuration $V$ and a transition sensor configuration $W$ such that $Q$ is structurally observable.
under $V$ and $L_W$, and $||V|| + ||W||$ is minimized.

Now we can construct an instance of the OPSS problem from this CGSS problem. Consider a partially observed Petri net $Q'$ with 5 observable transitions $t'_1, ..., t'_5$ corresponding to $t_1, ..., t_5$; 6 places $p''_1, p''_2, p'_1, p'_2, p'_3, p'_4$ in which $p''_1, p''_2$ correspond to the two types of transition sensors, $p'_1, p'_2, p'_3, p'_4$ correspond to $p_1, p_2, p_3, p_4$ in $Q$, and only $p'_4$ is unobservable; and labeling function defined as $L'(t'_1) = L'(t'_2) = L'(t'_3) = L'(t'_4) = L'(t'_5) = \lambda$. The incidence matrix $D' := \begin{bmatrix} U \\ D \end{bmatrix}$, where $D$ is the incidence matrix of $Q$ and

\[
U = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}
\]

as discussed in Example 4.11. Since there are no identically behaving transitions in $Q$, there are no identically behaving transitions in $Q'$ either, because $Q'$ is obtained by expanding $Q$ with additional observable places.

Now the goal in the constructed OPSS problem is to find a $V_{\text{min}}$ such that all transitions can be distinguished. For this example, it is easy to see that $Q$ being structurally observable under $V$ and $L_W$ is equivalent to $Q'$ being structurally observable under $V' := \begin{bmatrix} W \\ V \end{bmatrix}$ and $L'$.

In general, given an instance of the CGSS problem, one can construct an instance of the OPSS problem as shown in Algorithm 10. Now we briefly argue the correctness of the reduction. Note that for any place sensor configuration $V$ and any transition sensor configuration $W$ in $Q$, there is a place sensor configuration $V' := \begin{bmatrix} W \\ V \end{bmatrix}$ in $Q'$ because observable place $p''_i$ in $Q'$ (for $i = 1, ..., d$) corresponds to the type of transition sensors $T_i$ in $Q$ and observable place $p'_j$ in $Q'$ (for $j = 1, ..., n_1$) corresponds to observable place $p_j$ in $Q$, and vice versa. Also, $Q$ being structurally observable under $V$ and $L_W$ is equivalent to $Q'$ being structurally observable under $L'$ and $V' := \begin{bmatrix} W \\ V \end{bmatrix}$ because the labeling function essentially outputs nothing, and the construction of $V'$ provides the same distinguishability on transitions as the combination of $V$ and $W$. Most computation in the reduction comes from Step 1 which involves constructing the new partially observed Petri net $Q'$ and can be verified to have complexity $O(dm + nm)$. Therefore, the reduction is of
Algorithm 10 Reduction from CGSS to OPSS

**Input:** An instance of the CGSS problem as shown in Problem 4.5

**Output:** An instance of the OPSS problem as shown in Problem 4.1

1: Construct $Q'$: $T' = T'_o = \{t'_1, t'_2, ..., t'_m\}$, $P' = \{p''_1, p''_2, ..., p''_d, p'_1, p'_2, ..., p'_n\}$ of which only $p'_{n_1+1}, ..., p'_n$ are unobservable, $L'(t'_i) = \lambda$ for any $t'_i \in T'$, and $D' = \begin{bmatrix} U \\ D \end{bmatrix}$ where $D$ is the incidence matrix of $Q$, and $U(p''_i, t'_j) = 1$ (for $i = 1, ..., d$ and $j = 1, ..., m$) if $t_j$ is covered by $T_i$, and $U(p''_i, t'_j) = 0$ otherwise.

2: Output the OPSS problem instance with partially observed Petri net $Q'$ and fixed labeling function $L'$.

### 4.10 Summary

In this chapter, we studied optimal sensor selection problems to achieve structural observability in partially observed Petri nets. The place sensor selection problem was shown to be computationally hard. It can be solved (optimally) by transforming it into a BIP problem or (suboptimally) by employing approximation algorithms. We proposed four such algorithms — a top-down method, a bottom-up method, a combined method and an SCP based method — all of which have complexity that is polynomial in the number of places and transitions. Unlike the place sensor selection problem, the transition sensor selection problem was shown to be solvable with complexity that is polynomial in the number of places and transitions. However, the constrained OTSS problem (as well as the constrained general sensor selection problem) was shown to be convertible to the OPSS problem.

The approximation algorithms based on scoring functions can be easily adapted to the design of an immediate fault diagnoser [19], e.g., by slightly modifying our approximation algorithms to choose a set of places to put sensors on so that fault transitions (modeled as unobservable transitions in [19]) can be distinguished immediately. The heuristics proposed in this chapter can also be used to calculate reduct in rough set theory and approximate other $\mathcal{NP}$-complete problems (e.g., the minimum vertex cover problem) due
to the reduction in Section 4.4.
CHAPTER 5
APPLICATION EXAMPLES

In this chapter, we use two practical examples to demonstrate the results in previous chapters. The first example is a distribution network, which is used to demonstrate the polynomial bounds proposed in Chapter 3; the second example is a flexible manufacturing cell, which is used to illustrate approximation algorithms for the optimal place sensor selection problem.

5.1 Distribution Network

In this section, we consider the labeled Petri net model of a simplified distribution network as shown in Fig. 5.1(a): There is a distribution center that holds a certain number of units of a product (modeled by the number of tokens in place $p_3$) together with three retail stores that also hold certain numbers of units of product (modeled by the number of tokens in places $p_2$, $p_4$ and $p_6$). A large truck can deposit 2 units of product into the distribution center (this is modeled via transition $t_3$) while three smaller trucks can move 1 unit of product from the center to the corresponding retail stores (this is modeled via transitions $t_2$, $t_4$ and $t_6$). Transitions $t_1$, $t_5$ and $t_7$ model customers buying 1 unit of product from retail stores, whereas the number of tokens in places $p_1$, $p_5$ and $p_7$ captures the number of products customers have bought. The labeling function is $L(t_3) = a$, $L(t_1) = L(t_5) = L(t_7) = b$ and $L(t_2) = L(t_4) = L(t_6) = \lambda$. This labeling function implies that the distribution of products from the center to the retail stores is unobservable while the arrival of the huge truck generates label $a$ and customer purchases generate label $b$. The initial state is given as $M_0 = (0 \ 0 \ 5 \ 0 \ 0 \ 0 \ 0)^T$ (namely, there are five units of product in the distribution center and no product elsewhere) and our goal is to estimate (based on knowledge of the distribution network, its initial state, and the observation of labels) the state of the distribution...
network (i.e., the number of units of product available at each place). Later on, we also consider a different labeling function $L'$ which assumes that more sensing information is available.

Note that the unobservable subnet (shown in Fig. 5.1(b)) is acyclic and has no source transition (the only source transition is observable). Therefore, we can use Algorithm 11 in Appendix A to compute the vector $y_\lambda$. First, we use Eq. (A.1) in Appendix A to get the following partition of the unobservable subnet: $T_\lambda = T_1 \cup T_2$, where $T_1 = \emptyset$ and $T_2 = \{t_2, t_4, t_6\}$; $P_\lambda = P_1 \cup P_2$, where $P_1 = \{p_3\}$ and $P_2 = \{p_2, p_4, p_6\}$. Using the partition, we can rearrange the incidence matrix $D^\lambda_{\text{uo}}$ (that corresponds to the unobservable subnet) by exchanging the row corresponding to place $p_2$ with the row corresponding to place $p_3$; this results in the matrix

$$D' = \begin{pmatrix} -F_{1,2} \\ F_{2,2} \end{pmatrix},$$

where $F_{1,2} = [1 1 1]$ and $F_{2,2} = I_3$ ($I_3$ is the $3 \times 3$ identity matrix). To obtain $c_1$ and $c_2$ as defined in Corollary 3.1, we follow the algorithm: we start with $y'_\lambda = (1 1 1 1)^T$; then, after considering $t_6$, we get $y'_\lambda = (2 1 1 1)^T$; following the for loop in Algorithm 11, we get $y'_\lambda = (2 1 1 1)^T$ after considering $t_4$ and
After rearranging \( y'_\lambda \), we finally get \( y_\lambda = (1 \ 2 \ 1 \ 1)^T \) and, as a result, \( c_1 = 10 \) and \( c_2 = 4 \). As \( j = 1, \ell = 3 \) and \( l_\lambda = 3 \), the bound in Corollary 3.1 is

\[
\frac{(k + \frac{3}{2})^2}{((3 - 1)!)^2} \times \binom{10 + 4k + 3}{3} = \frac{(k + \frac{3}{2})^2}{2} \times \binom{13 + 4k}{3}.
\]

Now we compute the bound in Theorem 3.2. Since \( a_1 = 0, a_2 = 1, n_2 = 4, n = 7 \) and \( c_1 = 5, c_2 = 2 \) (using the vector \( y_\lambda = (1 \ 1 \ 1 \ 1)^T \)), the bound in Theorem 3.2 is

\[
(1 + 0 + k)^{7-4} \times (1 + 5 + 2k)^4 = (1 + k)^3 \times (6 + 2k)^4,
\]

which is worse than the bound obtained from Corollary 3.1 (because it is \( O(k^7) \) instead of \( O(k^5) \)).

If we have more sensors, the bound in Corollary 3.1 can be improved. For example, if transitions \( t_1, t_5 \) and \( t_7 \) have unique transition labels (i.e., if we are dealing with a new labeling function \( L' \) such that \( L'(t_3) = a, L'(t_1) = b, L'(t_5) = c, L'(t_7) = d \) and \( L'(t_2) = L'(t_4) = L'(t_6) = \lambda \)), then \( j = 0 \) and the bound in Corollary 3.1 becomes \( (13+4k)_3 \) while the bound in Theorem 3.2 does not improve. In order to validate our bounds against the actual number of consistent markings for the labeling function \( L' \), we randomly generate a sequence of labels \( \omega \) of length 15 with probability \( P(a) = \frac{1}{3} \) and \( P(b) = P(c) = P(d) = \frac{2}{9} \) and also use the fixed sequence \( aa...a \) of length 15 (which can be shown to generate the largest number of consistent markings among all observation sequences of length 15); then, we compute consistent markings using Algorithm 2. The actual numbers of consistent markings for these two sequences, as well as the upper bound obtained in Corollary 3.1, are plotted against the length of the observation sequence in Fig. 5.2.

### 5.2 Automated Guided Vehicles

In this section, we first consider a practical example to demonstrate the four approximation algorithms in Chapter 4 and then discuss how to modify our heuristics to solve the OPSS problem with arbitrary nonnegative integer

---

1For the definitions of \( j, \ell, l \) and \( l_\lambda \), refer to Corollary 3.1.
2For the definitions of \( a_1, a_2, n_2, c_1 \) and \( c_2 \), refer to Theorem 3.2.
5.2.1 Optimal Place Sensor Selection

In this subsection, we consider the OPSS problem in a flexible manufacturing cell (shown in Fig. 5.3(a)) modeled as a Petri net with 64 places and 53 transitions (shown in Fig. 5.3(b); the model was first introduced in [25]). We use this example to compare our approximation algorithms against the method based on binary integer programming solvers.

The cell includes three workstations, two part-receiving stations and one completed part station. Five automated guided vehicles (AGVs) transport material between pairs of stations and they may collide with each other in shared regions. In [25], the collision avoidance problem was studied under the assumption that the current marking of the system is known. The same problem was also studied in [76] under the assumption that all transitions are
Figure 5.3: A flexible manufacturing cell and its Petri net model.

observable. However, these methods cannot be directly applied when only part of the system state is known or when there are unobservable transitions, making the study of optimal sensor selection problems ideal for this setting.

We model the cell as a partially observed Petri net and assume that all 64 places and all 53 transitions are observable so that we do not need to worry about the existence of a solution to the OPSS problem. To test the effectiveness of our approximation methods for the OPSS problem, we need to generate labeling functions. First, we specify the number of transition labels $i$. We let $i$ take values 10, 13, 16, 20, 24, 30, and for each value of $i$, we randomly generate 5 labeling functions in the following manner: we allow each transition $t$ (among all 53 transitions) to have any of the $i$ labels with equal probability $\frac{1}{i}$. In total, we have 30 randomly generated labeling functions; then we solve the resulting 30 OPSS problems using the top-down method (as shown in Algorithm 5), the bottom-up method (as shown in Algorithm 6), the combined method (applying the top-down method after the bottom-up method), the SCP based method (as shown in Chapter 4.7), and the BIP based method (the solver we used is the open source mixed-integer programming solver [74]). Simulation programs were written in Matlab and were run on a 1.4 GHz laptop. The results obtained using the four approximation algorithms and the BIP based method are shown in Tables 5.1 and 5.2. In the tables, "$i$" refers to the number of transition labels, $q$ is de-
fined in Algorithm 7 (note that \( q \) captures the number of constraints in the BIP problem), \( OPT \ast H_q \) is the performance guarantee for the SCP based method as explained in Chapter 4.7. Note that the exhaustive search method is prohibitive for this example as there are \( 2^{64} = 1.8447 \times 10^{19} \) possibilities.

We compare the four approximation algorithms with the BIP based method in terms of running time and performance. The plot of running time for the four heuristic methods and the BIP based method is shown in Fig. 5.4. The plot shows that the four heuristic methods run much faster than the BIP based method especially when there are less transition sensors. Among these four heuristic methods, the SCP based method is the fastest.

In terms of performance, we first plot the number of sensors output by these five methods, and then compare the four heuristic methods with the BIP based method using the difference between the number of sensors in their respective solutions. The plot of the number of sensors output by the four heuristic methods and the BIP based method is shown in Fig. 5.5. The plot shows that the combined method performs best among all four heuristic methods. Though the SCP based method performs worst among all heuristic methods, the number of sensors generated by this method indeed satisfies the

Figure 5.4: Plot of running time.
Table 5.1: Simulation results of top-down method, bottom-up method, and combined method.

<table>
<thead>
<tr>
<th>i</th>
<th>Top-down Method</th>
<th>Bottom-up Method</th>
<th>Combined Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time (s)</td>
<td># sensors</td>
<td>time (s)</td>
</tr>
<tr>
<td>30</td>
<td>0.343</td>
<td>14</td>
<td>1.688</td>
</tr>
<tr>
<td></td>
<td>0.328</td>
<td>17</td>
<td>1.922</td>
</tr>
<tr>
<td></td>
<td>0.359</td>
<td>15</td>
<td>1.735</td>
</tr>
<tr>
<td></td>
<td>0.344</td>
<td>18</td>
<td>1.812</td>
</tr>
<tr>
<td></td>
<td>0.359</td>
<td>17</td>
<td>1.844</td>
</tr>
<tr>
<td>24</td>
<td>0.406</td>
<td>18</td>
<td>2.203</td>
</tr>
<tr>
<td></td>
<td>0.391</td>
<td>16</td>
<td>2.125</td>
</tr>
<tr>
<td></td>
<td>0.390</td>
<td>17</td>
<td>2.156</td>
</tr>
<tr>
<td></td>
<td>0.359</td>
<td>17</td>
<td>2.063</td>
</tr>
<tr>
<td></td>
<td>0.391</td>
<td>18</td>
<td>2.343</td>
</tr>
<tr>
<td>20</td>
<td>0.422</td>
<td>20</td>
<td>2.547</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>19</td>
<td>2.266</td>
</tr>
<tr>
<td></td>
<td>0.407</td>
<td>18</td>
<td>2.235</td>
</tr>
<tr>
<td></td>
<td>0.438</td>
<td>18</td>
<td>2.703</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>18</td>
<td>2.250</td>
</tr>
<tr>
<td>16</td>
<td>0.406</td>
<td>20</td>
<td>2.437</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>21</td>
<td>2.890</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>20</td>
<td>2.219</td>
</tr>
<tr>
<td></td>
<td>0.391</td>
<td>19</td>
<td>2.219</td>
</tr>
<tr>
<td></td>
<td>0.359</td>
<td>21</td>
<td>2.281</td>
</tr>
<tr>
<td>13</td>
<td>0.390</td>
<td>22</td>
<td>2.453</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>22</td>
<td>2.234</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>22</td>
<td>2.282</td>
</tr>
<tr>
<td></td>
<td>0.390</td>
<td>21</td>
<td>2.562</td>
</tr>
<tr>
<td></td>
<td>0.390</td>
<td>21</td>
<td>2.266</td>
</tr>
<tr>
<td>10</td>
<td>0.390</td>
<td>23</td>
<td>2.360</td>
</tr>
<tr>
<td></td>
<td>0.438</td>
<td>23</td>
<td>2.219</td>
</tr>
<tr>
<td></td>
<td>0.406</td>
<td>23</td>
<td>2.469</td>
</tr>
<tr>
<td></td>
<td>0.391</td>
<td>22</td>
<td>2.297</td>
</tr>
<tr>
<td></td>
<td>0.422</td>
<td>23</td>
<td>2.375</td>
</tr>
</tbody>
</table>
Table 5.2: Simulation results of SCP and BIP based methods.

<table>
<thead>
<tr>
<th>i</th>
<th>SCP based Method</th>
<th>BIP based Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time (s)</td>
<td># sensors</td>
</tr>
<tr>
<td>30</td>
<td>0.046</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>0.047</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>0.047</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>0.047</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>0.047</td>
<td>29</td>
</tr>
<tr>
<td>24</td>
<td>0.094</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>0.047</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>0.062</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>0.062</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>0.063</td>
<td>31</td>
</tr>
<tr>
<td>20</td>
<td>0.078</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>0.078</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>0.078</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>0.063</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>0.156</td>
<td>50</td>
</tr>
<tr>
<td>16</td>
<td>0.125</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>0.109</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>0.141</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>0.093</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>0.094</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>0.094</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>0.109</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>0.125</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>0.093</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>0.125</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>0.157</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>0.188</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>0.125</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>0.171</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>0.156</td>
<td>50</td>
</tr>
</tbody>
</table>
bound $OPT*H_q$ as shown in Fig. 5.5. Table 5.3 shows the comparison results when considering the difference $\Delta$ between the number of sensors given by heuristic methods and the number given by the BIP based method. The combined method has the best performance among all heuristic methods: 28 out of 30 simulations give a very close solution (namely, $\Delta \leq 1$). In addition, the combined method indeed performs at least as well as the bottom-up method, and improves the results for certain simulations. The SCP based method performs worst among all four heuristic methods (though it runs fastest among these four methods): only one simulation generates a place sensor configuration with 2 more sensors than the optimal solution, and all other simulations generate a much worse solution.

These simulations suggest that the top-down method, the bottom-up method, and the combined method run faster and can find reasonably good solutions compared with the BIP based method. In particular, the combined method is quite promising in terms of running time and quality of the approximation.
Table 5.3: Comparison of heuristic methods with BIP based method over 30 simulations.

<table>
<thead>
<tr>
<th>Δ</th>
<th>Top-down Method</th>
<th>Bottom-up Method</th>
<th>Combined Method</th>
<th>SCP based Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>13</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>≥3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>29</td>
</tr>
</tbody>
</table>

5.2.2 OPSS with Arbitrary Nonnegative Integer Costs

In this subsection, we consider a weighted version of the OPSS problem. More specifically, we associate with each observable place $p_i$ a nonnegative integer $\text{cost}(p_i)$ which captures the cost of a sensor on place $p_i$. Given a partially observed Petri net $Q$ and a fixed labeling function $L$, we try to find a valid place sensor configuration $V_{\text{min}}$ such that for any other valid sensor configuration $V$, $CTV_{\text{min}} \leq CTV$, where $C = (\text{cost}(p_1) \text{ cost}(p_2) \cdots \text{cost}(p_n))^T$.

The existence condition for an optimal solution is still given by Theorem 4.1 and can be proved in a way similar to that of Theorem 4.1. To solve the problem, we can transform it into a binary integer programming problem by setting the vector $c$ (in Chapter 4.5) to be $C$. Notice that binary integer programming solvers will give the optimal solution for this problem but will be slow for large problem instances. To employ the top-down method developed in Chapter 4.6, we use the following modified scoring function $f'(p_i) = \frac{f(p_i)}{\text{cost}(p_i)}$, where $f(p_i)$ is the scoring function defined in Chapter 4.6. The justification is the following:

(i) The larger the value of $f(p_i)$, the fewer place sensors are needed based on the result in Chapter 4.6.

(ii) The smaller the value of $\text{cost}(p_i)$, the smaller the total cost.

To use the bottom-up method, we can generalize $f'(p_i)$ to a set of places $S$ as $f'(S) = \frac{f(S)}{\sum_{p \in S} \text{cost}(p)}$. Other scoring functions with properties similar to (i) and (ii) above can also be used.

To compare the two modified approximation algorithms and the BIP based method for the OPSS problem with costs, we still use the AGV example. In our simulations, we randomly generate a labeling function with 20 labels and choose 5 cost functions: (i) function 1 has entries 0 or 1 with equal probability; (ii) function 2 (or 3, 4, 5) is a shifted version of function 1 by
Table 5.4: Simulation results for OPSS with costs.

<table>
<thead>
<tr>
<th>i</th>
<th>Top-down Method</th>
<th>Bottom-up Method</th>
<th>BIP based Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time (s) # sensors</td>
<td>time (s) # sensors</td>
<td>time (s) # sensors</td>
</tr>
<tr>
<td>1</td>
<td>0.391 18</td>
<td>2.938 38</td>
<td>0.125 23</td>
</tr>
<tr>
<td>2</td>
<td>0.375 18</td>
<td>2.297 36</td>
<td>1.781 19</td>
</tr>
<tr>
<td>3</td>
<td>0.390 18</td>
<td>2.109 21</td>
<td>5.250 17</td>
</tr>
<tr>
<td>4</td>
<td>0.375 18</td>
<td>2.015 19</td>
<td>13.266 17</td>
</tr>
<tr>
<td>5</td>
<td>0.391 18</td>
<td>2.000 18</td>
<td>14.296 17</td>
</tr>
</tbody>
</table>

(changing the expectation to be 1.5 (or 2.5, 5.5, 10.5). The results are shown in Table 5.4. In this table, “i” refers to the i-th cost function, and “cost” refers to the total cost of the corresponding place sensor configuration. The results show that the two approximation algorithms give solutions close to the optimal one but with much less running time, especially when the costs of places do not exhibit large relative difference. If we fix the cost function but change the labeling function (while keeping the total number of labels to be 20), the total cost and running time for all three methods do not change much, and we omit the outcomes of these simulations.)
CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

In this chapter, we summarize the work in this dissertation and identify potential future directions.

6.1 Conclusions

A variety of systems, such as manufacturing systems, computer networks, traffic systems, and others, can be modeled as discrete event systems at some level of abstraction. Discrete event models can then be used for the purpose of state estimation, supervisory control, and fault diagnosis. Knowledge of the system state is always critical for controller design and fault diagnosis. In this dissertation, we have studied two sensor related problems: state estimation and sensor selection for structural observability.

In the state estimation problem, we assume that only transition sensors are available (in the form of labeling functions) and consider two different types of uncertainty that arise due to limited sensors:

(a) Occurrence of distinct activities that generate the same observation.

(b) Occurrence of unobservable activities that go unrecorded.

We show that, under some reasonable assumptions about the given labeled Petri net’s unobservable subnet, the number of possible system states that are consistent with an observation sequence is upper bounded by a function that is polynomial in the length of the observation sequence. The implications of this observation include the following:

• State estimation problems of various forms can be solved in a very general setting with complexity that is polynomial in the length of the observation sequence.
• In many supervisory control and fault diagnosis applications that require state estimation, the Petri net model of the plant is fixed and only the number of observed labels increases with time. Therefore, broadly speaking, our bounds can also be used to argue that the computational complexity of supervisory control and fault diagnosis algorithms remains polynomial as long as the labeled Petri net model satisfies the assumptions in Chapter 3. One specific example is discussed in Chapter 3.6.

• The polynomial bounds can guide the design of systems, especially when configuring the state transition sensors, to reduce the uncertainty introduced in the state estimation stage.

• The polynomial bounds can also be used to establish the computational complexity of reachability checking for certain classes of Petri nets as shown in Chapter 3.7.

In the problem of sensor selection, both transition and place sensors are available. The goal of sensor selection is to place a minimum number of sensors so as to maintain the property of structural observability, i.e., the ability to uniquely determine the system state based on sensor information (and knowledge of the system model and initial state). To simplify the problem and gain a better understanding of it, we consider two subproblems: the optimal place sensor selection problem given a fixed set of transition sensors and the optimal transition sensor selection problem given a fixed set of place sensors. We establish that the existence of a solution to either the OPSS or the OTSS problem can be determined with complexity that is polynomial in the number of places and transitions of the Petri net. However, the OPSS problem itself is computationally hard though it can be solved (optimally) by transforming it into a BIP problem or (suboptimally) by employing approximation algorithms. We propose four polynomial approximation algorithms: the top-down method, the bottom-up method, the combined method, and the SCP based method. The example of automated guided vehicles shows that the four algorithms, especially the combined method, work almost as well as the BIP based method but with significantly less running time. On the other hand, the SCP based method has a provable performance guarantee. Unlike the OPSS problem, the OTSS problem is shown to be solvable
with complexity that is polynomial in the number of places and transitions of the given Petri net. We have also studied sensor selection problems in which there are constraints on the way transitions share sensors, and have shown that sensor selection problems with such constraints can be converted to the OPSS problem.

The approximation algorithms (based on scoring functions) for the OPSS problem could also be applied to other problems (e.g., sensor selection to achieve immediate diagnosis of faults [19], reduct calculation in rough set theory [20], approximating solutions for \(\mathcal{NP}\)-complete problems).

6.2 Future Directions

Besides the potential application of these results mentioned previously, there are several interesting directions for future exploration, including the following:

- Exploring the connections between sensor selection and identifying codes.
- Finding better ways to integrate estimation, diagnosis and control tasks.
- Examining issues of concurrency and unreliable observations arising in distributed settings.
- Addressing theoretical challenges in practical reachability algorithms.

All of the above challenges are discussed in more detail below.

Sensor Selection and Identifying Codes

The identifying code problem for a given undirected graph involves finding a minimum set of vertices such that any vertex in the undirected graph can be uniquely identified by examining the vertices (within this minimum set of vertices) that connect to it. The problem has been demonstrated to be fundamental in a wide variety of applications, including fault diagnosis and environmental monitoring; in addition, the problem has deep connections to information theory, superimposed and covering codes, and tilings [77]. Currently the identifying code problem is approximated by converting it into
a set cover problem and then utilizing known approximation algorithms for
the set cover problem. The OPSS problem is polynomially equivalent to the
identifying code problem because they are both equivalent to the set cover
problem (see [77] for the equivalence between the identifying code problem
and the set cover problem, and refer to Chapter 4.7 for the equivalence be-
tween the OPSS problem and the set cover problem). Simulations in Chap-
ter 5.2 show that our heuristic algorithms based on scoring functions for the
OPSS problem perform much better than the SCP based method. There-
fore, it is worthwhile to investigate the effectiveness of our heuristics to the
problem of devising new identifying codes.

Unified Framework for Estimation, Diagnosis and Control

Over the last decade, tremendous research progress has been accomplished
in the areas of state estimation, fault diagnosis and supervisory control of
complex discrete event systems. However, for the most part, these areas
have been addressed separately, and there is no unified framework in which
tasks such as estimation, diagnosis and control can be addressed simulta-
neously in a cohesive manner. Typically, when one tries to put different
techniques together, their underlying modeling frameworks and assumptions
pose problems to integration and it is important to have a carefully designed
architecture, so that such tasks can be accomplished while at the same time
existing techniques can be exploited to the greatest possible extent. The
work in this dissertation has individually addressed some of these tasks and
has the potential to be integrated and used as a basis for a unified framework.

Concurrency and Unreliable Observation

As discrete event systems become more distributed (even across geographi-
cal boundaries, e.g., the Internet), subsystems run concurrently and observa-
tions shared between subsystems could be unreliable due to sensor failures,
communication outages, or synchronization issues. Estimation, diagnosis
and control methods that are designed for centralized systems must be re-
formulated to address such challenging issues. The estimation and diagnosis
methods developed in this dissertation for centralized systems can poten-
tially be adapted to distributed settings by applying unfolding techniques
developed for model checking [78] to handle concurrency, and probabilistic settings to model unreliable observations.

Reachability Checking

Given a Petri net, the reachability problem asks if a final target state is reachable from a known initial state. The problem plays a central role in Petri net theory as a number of other problems (e.g., liveness analysis and deadlock checking) are recursively equivalent to it. Though the reachability problem has been shown to be decidable, its complexity is still an open problem. The study of the reachability problem involves research topics in graph theory, integer programming, formal languages and number theory. The work in Chapter 3.7 has shed some light on solving this problem, at least when certain algebraic conditions hold in the underlying Petri net. The study of the reachability problem for general Petri nets, perhaps by generalizing some of the ideas in this dissertation, is an exciting avenue for future research.

The above topics comprise some future research directions pertaining to this dissertation. In addition, it might be important to study the application of the results in this dissertation to monitoring, fault diagnosis and control of energy distribution networks, modeling and analysis of biological networks, and other settings.
We show that an acyclic Petri net $G$ without source transitions is deadlock structurally bounded by describing an algorithm for computing a vector $y$ with strictly positive integer entries such that $y^T D < 0^T_m$, where $D$ is the incident matrix of $G$ and $m$ is the number of transitions. This algorithm is useful as the polynomial bounds in Proposition 3.4 and Corollary 3.1 depend on the vector $y$.

For an acyclic Petri net $G$ without source transitions, we can define

$$P_k := \{ p \in P \setminus \bigcup_{i=1}^{k-1} P_i \mid p \subseteq \bigcup_{i=1}^k T_i \},$$

$$T_{k+1} := \{ t \in T \setminus \bigcup_{i=1}^k T_i \mid t \subseteq \bigcup_{i=1}^k P_i \} \quad \text{(A.1)}$$

for $k \in \{1, 2, 3, \ldots\}$ (see Chapter 5 in [66]). We mention the following properties of this partition. Since the number of transitions (or the number of places) is finite, there exists a positive integer $\mu$ (or $\mu'$) such that $T_k \neq \emptyset$ if $1 < k \leq \mu$ and $T_k = \emptyset$ if $k > \mu$ (or $P_k \neq \emptyset$ if $1 \leq k \leq \mu'$ and $P_k = \emptyset$ if $k > \mu'$); therefore, the set of transitions $T$ is partitioned into $\mu - 1$ nonempty sets and the set of places $P$ is partitioned into $\mu'$ nonempty sets, where $\mu = \mu'$ or $\mu = \mu' + 1$. For simplicity, we assume that $\mu = \mu'$; if $\mu = \mu' + 1$, it is straightforward to modify our algorithm to accommodate this case. With this partition, we can order rows of the incidence matrix $D$ such that places in $P_1$ come first, then places in $P_2$, and so forth; we can also order columns of the incidence matrix $D$ such that transitions in $T_2$ come first, then transitions
Algorithm 11 Computation of Vector $y$

Input: An acyclic Petri net $G$ without source transitions

Output: A vector $y$ with strictly positive integer entries such that $y^T D < 0^T_m$

1: Rearrange the rows and columns of $D$ to obtain $D'$ and keep the mapping of rows;
2: Let $y' = 1^n$;
3: for $j = m, m - 1, \ldots, 1$ do
4:   if $y'^T D'(\cdot, j) < 0$ then
5:     go to the next $j$;
6:   else
7:     choose the smallest index $i$ such that $D'(i, j) < 0$ and increase the value $y'(i)$ to the smallest integer such that $y'^T D'(\cdot, j) < 0$;
8: end if
9: end for
10: Rearrange the vector $y'$ using the mapping of rows to get the $y$ corresponding to $D$;
11: Output the vector $y$.

in $T_3$, and so forth. Now $D$ has the following block structure:

$$D' = \begin{pmatrix}
-F_{1,2} & -F_{1,3} & \cdots & -F_{1,\mu} \\
F_{2,2} & -F_{2,3} & \cdots & -F_{2,\mu} \\
& \vdots & \ddots & \vdots \\
F_{\mu-1,2} & F_{\mu-1,3} & \cdots & -F_{\mu-1,\mu} \\
F_{\mu,2} & F_{\mu,3} & \cdots & F_{\mu,\mu}
\end{pmatrix}, \quad (A.2)$$

where $F_{i,j}$ is a $|P_i| \times |T_j|$ matrix with nonnegative integer entries for $i = 1, \ldots, \mu$ and $j = 2, \ldots, \mu$. These properties and the block structure of $D'$ are proved in [66] for the general case without the assumption that there are no source transitions.

One observation about the block structure of $D'$ is that as long as there is a block $-F_{i,j}$ for some $1 \leq i \leq \mu$ and $2 \leq j \leq \mu$, then all the blocks on the right of $-F_{i,j}$ have the form $-F_{i,j'}$ for $j' = j + 1, \ldots, \mu$. Now we describe Algorithm 11 to compute a vector $y$ such that $y^T D < 0^T_m$.

In the algorithm, we can always find the smallest index $i$ such that $D'(i, j) < 0$ because there is no source transition in the Petri net. The key idea of this algorithm is that when we increase the $i$-th entry of $y'$ for some $j$, $y'^T D'(\cdot, j') < 0$
holds for \( j' = j + 1, \ldots, m \):

(i) If \( j \) and \( j' \) belong to the same column block, it is obvious.

(ii) If \( j \) belongs to a column block on the left of the block that \( j' \) belongs to, the result holds because in \( D' \), \( D'(i, j') \leq 0 \) as long as \( D'(i, j) < 0 \) (due to the structure of \( D' \)).

This implies that the vector \( y \) obtained from Algorithm 11 indeed satisfies \( y^T D < 0^T_m \).
APPENDIX B

EQUIVALENCE BETWEEN STRUCTURAL OBSERVABILITY AND $K$-DELAYED STRUCTURAL OBSERVABILITY

**Proposition** Given a place sensor configuration $V$ and a labeling function $L$, a partially observed Petri net $Q$ being structurally observable is equivalent to $Q$ being $K$-delayed structurally observable.

**Proof:** Structural observability $\implies K$-delayed structural observability. If structural observability holds, $K$-delayed structural observability also holds by definition.

$K$-delayed structural observability $\implies$ structural observability. Suppose $K \geq 1$ (the case for $K = 0$ is trivial). If $K$-delayed structural observability holds, then the system state at time step $i$ may not be determined uniquely based on the observation sequence generated by transitions up to time step $i$ but can be determined after no more than $K$ additional transition firings. In other words, some of the possible states at time step $i$ given the observations up to time step $i$ vanish due to lack of tokens in certain places (i.e., there are not enough tokens to enable firing sequences up to time step $i + K$). However, these states will still be possible even after $K$ time steps if enough tokens are added in the initial marking while generating the same observation. Since the property has to hold for an arbitrary initial state, we have reached a contradiction. Therefore, the system state at time step $i$ has to be uniquely determined by observations up to time step $i$, which implies that structural observability holds.
APPENDIX C

COMPUTATIONAL COMPLEXITY OF ALGORITHMS 5 AND 6

**Proposition** Both Algorithm 5 and Algorithm 6 have complexity $O(n^2 m^2)$ that is polynomial in the number of places and transitions.

**Proof:** Specifically, Line 1 of Algorithm 5 (or Line 1 of Algorithm 6) has complexity $O(n m^2)$ following Proposition 4.3. Line 2 of Algorithm 5 (or Line 2 of Algorithm 6) has complexity $O(n m)$ because we copy the Petri net and add another isolated transition if there is at least one unobservable transition. Line 4 and Line 12 of Algorithm 5 (or Line 3, Line 4 and Line 10 of Algorithm 6) have complexity $O(n)$ as we do the assignment of $n_1$-dimensional vectors ($n_1 \leq n$).

Line 3 of Algorithm 5 computes partitions generated by every observable place $p_i$ and has complexity of roughly $n \sum_{e:|Te|\geq 2} (|Te|^2 + 1) = O(nm^2)$, where $n$ refers to at most $n$ observable places, $|Te|^2$ refers to the complexity of computing the partition of $Te$, 1 refers to the summation needed to calculate $f(p_i)$, and the term $O(nm^2)$ can be derived using the fact that $\sum_{i=1}^{n} a_i^2 \leq (\sum_{i=1}^{n} a_i)^2$ for positive $a_i$'s. Lines 5-11 of Algorithm 5 have complexity $O(n(n + nm^2))$ (i.e., $O(n^2 m^2)$), where the first $n$ refers to subtracting at most $n$ places, the second $n$ refers to the assignment of vectors and the searching of $p$ in $P_{left}$ to minimize $f(p)$, and $nm^2$ refers to the computational complexity of testing transition distinguishability. Putting all steps together, we conclude that the total complexity of Algorithm 5 is $O(n^2 m^2)$.

Lines 5-9 of Algorithm 6 have complexity $O(n(nm^2 + n))$ (i.e., $O(n^2 m^2)$), where the first $n$ refers to adding at most $n$ places, $nm^2$ refers to the computation for finding $p$ in $P_{left}$ to maximize $f(S \cup \{p\})$ (the argument is similar to that of Line 3 of Algorithm 5), and the last $n$ refers to the assignment of $n_1$-dimensional vectors ($n_1 \leq n$).

---

1We can compute the partition of $Te$ generated by place $p_i$ in the following way: (i) we first pick one transition $t \in Te$; (ii) we then search for another transition $t' \in Te$ such that $D(p_i, t) = D(p_i, t')$ which can be done in at most $|Te|$ steps; (iii) finally, we remove all transitions selected in Steps (i) and (ii), and repeat (i) and (ii) for at most $|Te|$ times. Overall, the computation of the partition has complexity $O(|Te|^2)$. 

117
vectors. Therefore, the total complexity of Algorithm 6 is also $O(n^2m^2)$. ■
Proposition Algorithm 8 has complexity $\mathcal{O}(nm^2)$ that is polynomial in the number of places and transitions.

Proof: Line 1 of Algorithm 8 has complexity $\mathcal{O}(nm^2)$ following Proposition 4.3. Line 2 of Algorithm 8 computes the partition of $T_o$ generated by $V$ and has complexity $\mathcal{O}(nm^2)$ because it is essentially the same as computing the partition of $T_e$ generated by place $p_i$ (other than the fact that we need to compare vectors with at most $n$ entries instead of comparing scalars) as shown in Appendix C. Line 3 of Algorithm 8 has complexity $\mathcal{O}(m)$ as we take the max operation for at most $m$ numbers. Lines 4-12 of Algorithm 8 have complexity $\mathcal{O}(m)$ as we only need to assign one label to each observable transition. Putting all steps together, we conclude that the total complexity of Algorithm 8 is $\mathcal{O}(nm^2)$. ■
REFERENCES


<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ-free labeled Petri net</td>
<td>12</td>
</tr>
<tr>
<td>λ-free labeling function</td>
<td>12</td>
</tr>
<tr>
<td>( P )</td>
<td>15</td>
</tr>
<tr>
<td>( P )-complete</td>
<td>15</td>
</tr>
<tr>
<td>( P )-hard</td>
<td>15</td>
</tr>
<tr>
<td>arithmetic-mean/geometric-mean inequality</td>
<td>26</td>
</tr>
<tr>
<td>acyclic Petri net</td>
<td>8</td>
</tr>
<tr>
<td>algebraic coding technique</td>
<td>40</td>
</tr>
<tr>
<td>alphabet</td>
<td>12</td>
</tr>
<tr>
<td>arc</td>
<td>6</td>
</tr>
<tr>
<td>automata</td>
<td>2</td>
</tr>
<tr>
<td>backward conflict-free</td>
<td>32</td>
</tr>
<tr>
<td>basis marking</td>
<td>40</td>
</tr>
<tr>
<td>basis reachability tree</td>
<td>40</td>
</tr>
<tr>
<td>belief</td>
<td>40</td>
</tr>
<tr>
<td>belief on the normal operation</td>
<td>42</td>
</tr>
<tr>
<td>belief on the occurrence of faults of type ( F_i )</td>
<td>42</td>
</tr>
<tr>
<td>binary integer programming</td>
<td>5</td>
</tr>
<tr>
<td>BIP</td>
<td>16</td>
</tr>
<tr>
<td>block structure</td>
<td>114</td>
</tr>
<tr>
<td>bottom-up method</td>
<td>80</td>
</tr>
<tr>
<td>bounded</td>
<td>8</td>
</tr>
<tr>
<td>ceiling function</td>
<td>54</td>
</tr>
<tr>
<td>CGSS</td>
<td>90</td>
</tr>
<tr>
<td>combined method</td>
<td>82</td>
</tr>
<tr>
<td>consistent firing sequence</td>
<td>41</td>
</tr>
<tr>
<td>consistent marking</td>
<td>10, 12, 41</td>
</tr>
<tr>
<td>constrained general sensor selection</td>
<td>90</td>
</tr>
<tr>
<td>constrained optimal transition sensor selection</td>
<td>90</td>
</tr>
<tr>
<td>contact-free</td>
<td>32</td>
</tr>
<tr>
<td>contact-free assumption</td>
<td>18</td>
</tr>
<tr>
<td>control set</td>
<td>11</td>
</tr>
<tr>
<td>control value</td>
<td>11</td>
</tr>
<tr>
<td>control-disabled</td>
<td>11</td>
</tr>
<tr>
<td>controllable transition</td>
<td>9</td>
</tr>
<tr>
<td>COTSS</td>
<td>90</td>
</tr>
<tr>
<td>deadlock structurally bounded Petri net</td>
<td>29</td>
</tr>
<tr>
<td>decision problem</td>
<td>15</td>
</tr>
<tr>
<td>DES</td>
<td>1</td>
</tr>
<tr>
<td>deterministic label</td>
<td>12</td>
</tr>
<tr>
<td>deterministic transition</td>
<td>12</td>
</tr>
<tr>
<td>diagnostic probe</td>
<td>76</td>
</tr>
<tr>
<td>discrete event</td>
<td>1</td>
</tr>
<tr>
<td>discrete event system</td>
<td>1</td>
</tr>
<tr>
<td>distribution network</td>
<td>97</td>
</tr>
<tr>
<td>edge</td>
<td>15</td>
</tr>
<tr>
<td>empty label</td>
<td>12</td>
</tr>
<tr>
<td>enabled transition</td>
<td>11</td>
</tr>
<tr>
<td>event-detectable</td>
<td>66</td>
</tr>
<tr>
<td>existence condition for OPSS</td>
<td>69</td>
</tr>
<tr>
<td>existence condition for OTSS</td>
<td>71</td>
</tr>
<tr>
<td>extended reduced Luenberger observer</td>
<td>32</td>
</tr>
<tr>
<td>fault diagnosis</td>
<td>4, 17</td>
</tr>
<tr>
<td>finite state machine</td>
<td>2</td>
</tr>
<tr>
<td>firing sequence</td>
<td>7</td>
</tr>
<tr>
<td>firing vector</td>
<td>7</td>
</tr>
<tr>
<td>floor function</td>
<td>54</td>
</tr>
<tr>
<td>general sensor selection problem</td>
<td>64</td>
</tr>
<tr>
<td>genetic algorithm</td>
<td>61</td>
</tr>
</tbody>
</table>
hybrid system, 1
identically behaving transitions, 66
immediate observability, 65
incidence matrix, 7
input dominant Petri net, 28
interpreted Petri net, 40, 61
invertibility, 66

L'Hôpital's Rule, 25
labeled Petri net, 12
labeling function, 10, 12

marking, 7
minimal diagnoser, 40
minimal sensor configuration, 60
minimum vertex cover, 5
minimum vertex cover problem, 15
monotone Petri net, 53
multiple faults, 51

net unfolding technique, 40
nondeterministic label, 12
nondeterministic transition, 12

observable place, 9
observable transition, 9, 12
observation sequence, 12
OPSS, 4, 65
optimal place sensor selection, 4, 62, 65
optimal transition sensor selection, 4, 62, 65
OTSS, 4, 65

partially observed and partially controlled Petri net, 9
partition of $T$ generated by $V$, 91
partition of $T_o$ generated by observable place, 77
partition of $T_o$ generated by $V$, 85
Petri net, 2, 7
Petri net structure, 6
place, 3, 6
place sensor, 3
place sensor configuration, 9

polynomial algorithm for OTSS, 87
polynomial bound, 26, 31
polynomial reduction, 4
polynomial time algorithm, 15
programmable logic controller, 3
reachability checking, 4
reachability problem, 52, 53
reachability set, 8
reachability tree, 52
reachable, 7
reduct, 5
reduction, 15
Reduction from CGSS to OPSS, 95
Reduction from COTSS to OPSS, 93
reduction from OPSS to SCP, 84
repeated faults, 51
rough set theory, 5

safe Petri net, 8
safety-critical system, 62
scoring function, 78
SCP, 16
sequence estimation, 4
source transition, 6
state equation, 7
state estimation, 32
state-enabled, 7
stochastic abstraction, 2
strictly monotone Petri net, 53
structural marking observability, 65
structural observability, 4, 61
structurally bounded, 8
structurally observable, 64
supervisor synthesis, 4
supervisory control, 17
switched system, 1
time complexity function, 15
time step, 61
timed abstraction, 2
token, 3, 7
top-down method, 79
transition, 3, 6
transition distinguishable, 66
transition firing, 7  
transition invariant, 7  
transition sensor, 3  
transition sensor configuration, 89  
uncontrollable subnet, 11  
uncontrollable transition, 10  
undirected graph, 15  
unobservable place, 9  
unobservable reach, 38  
unobservable subnet, 11, 13  
unobservable transition, 10, 12  
untimed abstraction, 1  
upper bound, 17  
upper bound for $\lambda$-free labeled Petri nets, 25  
upper bound for general labeled Petri nets, 28, 30, 31  
valid labeling function, 65  
valid place sensor configuration, 65  
vertex, 15  
vertex cover, 16  
weight function, 6, 41, 43