



HOKKAIDO UNIVERSITY

Title	Model Reduction and Control of Boolean Networks Using Binary Decision Diagrams
Author(s)	本山, 風馬
Degree Grantor	北海道大学
Degree Name	博士(情報科学)
Dissertation Number	甲第16518号
Issue Date	2025-03-25
DOI	https://doi.org/10.14943/doctoral.k16518
Doc URL	https://hdl.handle.net/2115/95207
Type	doctoral thesis
File Information	Fuma_Motoyama.pdf



SSI-DT46225023

Doctoral Dissertation

**Model Reduction and Control of Boolean Networks
Using Binary Decision Diagrams**

Fuma Motoyama



December, 2024

Course of Systems Science and Informatics
Graduate School of Information Science and Technology
Hokkaido University

Doctoral Dissertation
submitted to Graduate School of Information Science and Technology,
Hokkaido University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy.

Fuma Motoyama

Dissertation Committee: Professor Koichi Kobayashi (Chief examiner)
Professor Yuh Yamashita
Professor Satoshi Kanai

Copyright © 2024 by Fuma Motoyama. All rights reserved.

Model Reduction and Control of Boolean Networks Using Binary Decision Diagrams*

Fuma Motoyama

Abstract

A BN is well known as a discrete model for analysis and control of complex networks such as gene regulatory networks. In the last decade, there has been much attention on analysis and control of BNs. There, the semi-tensor product (STP) method is widely used for calculations on BN. However, STP has a weakness in that it cannot represent large BNs. Since complex networks are large-scale in general, it is important to consider the model reduction. In this thesis, a model reduction method and control method of BNs using a BDD (binary decision diagram) is proposed. A BDD is a data structure used to represent Boolean functions. Logical operations of Boolean functions using a BDD are more efficient than using the STP method. The organization of this thesis is as follows.

In Chapter 1, we illustrate this study's background and the motivation.

In Chapter 2, we explain BNs, BDDs, and the existing BN reduction method in preparation for the main topics in Chapters 2 to 4. We explain interaction graphs, transition graphs, and fixed points as important elements of BNs. In explaining the existing BN reduction method, we provide examples of calculations using semi-tensors and show that this method cannot be applied to large-scale BNs.

In Chapter 3, we consider implementing the existing reduction method using BDDs. Since the calculations of the existing method cannot be simply replicated using BDDs, it is necessary to redevelop the procedures that assume the use of BDDs. In order to use BDDs efficiently, we consider utilizing the interaction graph expressing interactions between elements of the state. We show that this method is effective in large-scale BNs where STP method is not applicable.

The reduction method in Chapter 3 does not pursue minimality. In Chapter 4, we consider reduction method to further reduce the reduced model obtained in Chapter 3. We construct a reduction policy that is different from the existing reduction method that is used as the basis in Chapter 3. Like the method in Chapter 3, this method makes it possible to perform calculations using BDDs by utilizing interaction graphs. We show that this method is effective for BNs that cannot be further reduced by the method in Chapter 3.

In Chapter 5, we consider the finite-time control problem of BNs. The finite-time control problem is to find all sequences of initial states and control input sequences that can reach the target state at a given final time. This problem has been proven to be NP-hard. In this chapter, we present a method for solving finite-time control problems using BDDs. Although BDDs have worse computational complexity in the worst case, they provide efficient solutions in many cases. In addition, We consider the problem of

*Doctoral Dissertation, Course of Systems Science and Informatics, Graduate School of Information Science and Technology, Hokkaido University, SSI-DT46225023, December 5, 2024.

applying some control specifications to a typical finite-time control problem. We show an example of calculation to find all solutions for a large-scale BN, and an example to extract solutions that satisfy the control specification from that solutions.

In Chapter 6, this thesis is concluded with a summary of the obtained results and some perspectives on the open problems.

Keywords: Boolean network (BN), binary decision diagram (BDD), model reduction, finit-time control, systems biology.

Contents

1	Introduction	1
1.1	Background	1
1.2	Contributions and Organization of This Thesis	2
2	Preliminaries	5
2.1	Notation	5
2.2	Boolean Networks	5
2.3	Boolean Networks with control inputs	7
2.4	Binary Decision Diagrams	9
2.5	FVS-Based Model Reduction of Boolean Networks	12
2.5.1	Overview	12
2.5.2	Model Reduction Using the Semi-Tensor Product Method	15
3	BDD-Based Model Reduction	17
3.1	Introduction	17
3.2	BDD-Based Implementation Method	17
3.2.1	Technical Issue of BDDs in Model Reduction	18
3.2.2	Proposed Method	18
3.3	Numerical Examples	23
3.4	Conclusion	25
4	Model Reduction Based on Complement and Absorption Laws	29
4.1	Introduction	29
4.2	Proposed Method	29
4.2.1	Motivating Example	30
4.2.2	Proposed Model Reduction Method	30
4.3	Numerical Example	36
4.4	Conclusion	37
5	BDD-Based Finite-time Control	39
5.1	Introduction	39
5.2	Problem Formulation	39
5.3	Proposed Solution Method	40
5.3.1	Solution Method Using BDDs	40
5.3.2	Example	41
5.4	Numerical Example	42

5.5	Control problem that satisfies control specifications	45
5.5.1	Problem Formulation	45
5.5.2	Solution	46
5.5.3	Example	48
5.6	Conclusion	50
6	Conclusions	51
	Acknowledgements	53
	References	55

List of Figures

2.1	Interaction graph in Example 1.	6
2.2	Transition graph in Example 1.	7
2.3	Interaction graph in Example 4.	8
2.4	Transition graph in Example 2.4.	10
2.5	Transition graph in Example 2.4 that applied controller 2.5.	11
2.6	The procedure to generate the logical expression $a \wedge b \vee \neg c$ by Apply operation.	12
2.7	Interaction graph in Example 6.	14
3.1	Flowchart of the proposed procedure.	20
3.2	Parent vertices connected to the vertices in $c_o = [3, 6, 2]$	21
3.3	Reduction of $\hat{f}_2(\cdot)$ based on Fig. 3.2, where $\hat{\cdot}$ is omitted.	22
3.4	Interaction graph in Example 10.	24
3.5	Parent vertices connected to the vertices in c_o in Example 10.	26
3.6	Interaction graph for the reduced BN in Example 10.	27
4.1	Interaction graph of the BN (4.4).	33
4.2	Interaction graph of the BN (4.6).	36
5.1	BDD of $F(x(0), U)$ in the example in Section 5.3.2.	42
5.2	Interaction graph in the example in Section 5.4.	44
5.3	Time response of the Hamming distance.	45
5.4	BDD of $F(U)$ in the example in Section 5.5.3.	48

List of Tables

3.1	Value of the variable after each loop	21
5.1	The computation result and the computation time for each T	43

Chapter 1

Introduction

1.1 Background

A gene regulatory network is a network that represents the interactions between molecules involved in gene expression. In recent years, improvements in gene expression measurement techniques, such as microarrays, have made it easier to obtain comprehensive expression data than before, and as a next step, research is being conducted to understand the dynamics. In order to understand the dynamics of molecular interactions, it is effective to introduce engineering knowledge about complex networks. Gene regulatory network models are broadly divided into continuous models and discrete models. In the continuous model, ordinary differential equation is used to describe the asymptotic behavior of molecules (see, e.g., [1, 2, 3]). The discrete model represents a continuous model in a simple form. So far, modeling using Boolean networks[4, 5], Bayesian networks[6], Petri nets[7, 8], etc. have been proposed. Although discrete models have lower expressive performance than continuous models, they are useful for understanding the general laws of a system.

A BN plays an important role in various applications of gene regulatory networks [9, 10], fault diagnosis [11, 12], and many other areas [13, 14]. In a BN, the state and the control input take a binary value, and time evolution of the state is represented by a Boolean functions. There is a weakness that a BN is too simple, but a BN is widely used as the first step of developing control theory of complex networks. Many fundamental results have been obtained, such as stability [15, 16, 17, 18], stabilization [19, 20, 21], controllability [22, 23, 24], observability [22, 25, 26], and optimal control [27, 28, 29, 30]. To model more complex behavior, a probabilistic Boolean network (PBN) [31, 32] and context-sensitive PBN [33] have been proposed as an extended model of BNs. For also these extended models, fundamental results [34, 35, 36, 37] have been obtained.

A control problem of BNs aims to obtain an intervention strategy to steer the target network to a desired state. This technology could be used for developing systematic drug discovery and cancer treatment[38, 39]. In the last decade, the semi-tensor product (STP) method has been widely used in analysis and control of BNs (see, e.g., [22] and [40]). Using the STP method, a given BN is equivalently transformed into a linear algebraic representation. Hence, analysis/control problems can be solved in an algebraic way. However, the STP method has a weakness. For a BN with n nodes and m control inputs, matrices of the size $2^n \times 2^{n+m}$ must be handled. From this fact, BNs such that the STP method can be applied are limited. Since gene networks are generally large, it is highly important to

develop control methods for large-scale BNs.

In analysis and control of large-scale complex networks, it is important to appropriately simplify a mathematical model. Several model reduction methods for complex networks have been obtained (see, e.g., [41, 42, 43]). For BNs, some results have been obtained [44, 45, 46, 47, 48]. In [44], the model reduction method using l_1 -gain analysis has been proposed. In [45, 46, 47, 48], the model reduction methods that the information on singleton attractors (fixed points) is preserved have been proposed. In the model reduction of BNs, an implementation method has not been studied. Using the STP method, we can implement the model reduction method. [44, 48] use the STP method, but this makes the method inapplicable to large networks. Thus, despite the model reduction is one of the fundamental problems in systems and control theory, a computationally efficient method for the model reduction of BNs has not been proposed.

A BDD (binary decision diagram) and SAT (satisfiability)/SMT (satisfiability modulo theories) solvers are well known as efficient tool for handling Boolean functions. A BDD is a data structure used to represent Boolean functions [49]. An efficient algorithm for performing logical operations (AND, OR, XOR, and so on) on BDDs has been proposed. A BDD efficiently works analysis of BNs, such as attractor detection [50, 51]. The problem of finding attractors using computational tools has been widely studied. For example, in [52], a solution method using integer programming has been proposed. SAT/SMT solvers determine whether a given Boolean function is satisfiable or not. The Yices SMT solver (<https://yices.csl.sri.com/>), the Z3 Theorem Prover (<https://github.com/Z3Prover>) and so on have been developed. See also the review paper [53]. SAT/SMT solvers have been used for attractor detection [54] and design of BN based on attractors [55]. To the best of our knowledge, BDDs and SAT/SMT solvers have not been applied to the reduction and control problems of BNs.

1.2 Contributions and Organization of This Thesis

In this thesis, we utilize BDD to enable the reduction and control of large-scale BNs. The main contributions of this thesis are:

1. To implement the reduction method of [45, 47] using BDD.
2. To propose a reduction method using BDD to further reduce the reduced BN obtained by method of [45, 47].
3. To propose a finite-time control method for BN using BDD.

The organization of this thesis is as follows.

In Chapter 2, we explain BNs, BDDs, and the existing BN reduction method in preparation for the main topics in Chapters 2 to 4. In explaining the existing BN reduction method, we provide examples of calculations using semi-tensors and show that this method cannot be applied to large-scale BNs.

In Chapter 3, we consider implementing the existing reduction method [45, 47] using BDDs. Since the calculations of this method cannot be directly replaced by BDDs, it is necessary to redevelop the procedures that assume the use of BDDs.

The reduction method in Chapter 3 does not pursue minimality. In Chapter 4, we consider reduction method to further reduce the reduced model obtained in Chapter 3.

Although this method is inefficient in terms of reduction performance and the amount of calculation, it is effective when combined with the reduction method in Chapter 3.

In Chapter 5, we consider the finite-time control problem of BNs. The finite-time control problem is to find all sequences of initial states and control input sequences that can reach the target state at a given final time. This problem has been proven to be NP-hard in [56]. In this chapter, we present a method for solving finite-time control problems using BDDs. Although BDDs have worse computational complexity in the worst case, they provide efficient solutions in many cases. In addition, We consider the problem of applying some control specifications to a typical finite-time control problem. In this chapter, we show a calculation example to extract a solution that satisfies this control specification from the solutions obtained by this method.

In Chapter 6, this thesis is concluded with a summary of the obtained results and some perspectives on the open problems.

Chapter 2

Preliminaries

2.1 Notation

Let $\{0, 1\}^n$ denote the set of n -dimensional vectors, which consists of elements 0 and 1. For the numbers $x_1, x_2, \dots, x_n$ and the index set $\mathcal{I} = \{i_1, i_2, \dots, i_m\} \subseteq \{1, 2, \dots, n\}$, define $[x_i]_{i \in \mathcal{I}} := [x_{i_1}, x_{i_2}, \dots, x_{i_m}]^\top$.

2.2 Boolean Networks

Consider the following BN:

$$\begin{cases} x_1(t+1) = f_1([x_j(t)]_{j \in \mathcal{N}_1}), \\ x_2(t+1) = f_2([x_j(t)]_{j \in \mathcal{N}_2}), \\ \vdots \\ x_n(t+1) = f_n([x_j(t)]_{j \in \mathcal{N}_n}), \end{cases} \quad (2.1)$$

where $x = [x_1, x_2, \dots, x_n]^\top \in \{0, 1\}^n$ is the state (e.g., expression level of the gene), and $t = 0, 1, 2, \dots$ is the discrete time. The function $f_i : \{0, 1\}^{|\mathcal{N}_i|} \rightarrow \{0, 1\}$ is a given Boolean function using logical operators such as logical AND ($\wedge$), logical OR ($\vee$), and logical NOT ($\neg$). The index set $\mathcal{N}_i$ is the set of states used in the function f_i . If the index set $\mathcal{N}_i$ is an empty set, $x_i(t+1)$ is uniquely determined to be 0 or 1.

Example 1. Consider the following BN model for a simplified apoptotic network [57]:

$$\begin{cases} x_1(t+1) = x_1(t), \\ x_2(t+1) = x_1(t) \wedge \neg x_3(t), \\ x_3(t+1) = \neg x_2(t) \wedge x_4(t), \\ x_4(t+1) = x_1(t) \vee x_3(t), \end{cases} \quad (2.2)$$

where

- x_1 is the concentration level (high or low) of the tumor necrosis factor (TNF, a stimulus),
- x_2 is the concentration level of the inhibitor of apoptosis proteins (IAP),
- x_3 is the concentration level of the active caspase 3 (C3a),

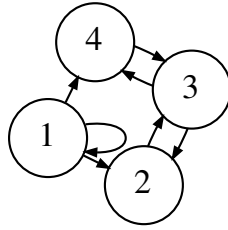


Figure 2.1: Interaction graph in Example 1.

- x_4 is the concentration level of the active caspase 8 (C8a).

See, e.g., [58] for more complicated models of an apoptosis network. In this BN model, $\mathcal{N}_1 = \{1\}$, $\mathcal{N}_2 = \{1, 3\}$, $\mathcal{N}_3 = \{2, 4\}$, and $\mathcal{N}_4 = \{1, 3\}$ hold.

□

The interaction between states in the BN can be represented by an interaction graph defined as follows.

Definition 2. The interaction graph is given by the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \dots, n\}$ is the set of vertices and $\mathcal{E} = \{(j, i) \in \mathcal{V} \times \mathcal{V} \mid j \in \mathcal{N}_i\}$ is the set of arcs.

Each vertex of the interaction graph corresponds to each state of the BN. When the state x_j affects the value of the function f_i , the directed edge is connected from the vertex x_j to the vertex x_i . The interaction graph of the BN in Example 1 is shown in Fig. 2.1. The sets $\mathcal{V}$ and $\mathcal{E}$ in this interaction graph are given by

$$\begin{aligned}\mathcal{V} &= \{1, 2, 3, 4\}, \\ \mathcal{E} &= \{(1, 1), (1, 2), (1, 4), (2, 3), (3, 2), (3, 4), (4, 3)\},\end{aligned}$$

respectively.

The state transition in the BN can be represented by a transition graph. Each vertex of the transition graph corresponds to a possible state of the BN. When the state x^A transition to x^B , the directed edge is connected from the vertex x^A to the vertex x^B . One important property of a transition diagram is that a vertex has exactly one outgoing edge. The transition graph of the BN in Example 1 is shown in Fig. 2.2.

Next, we define a fixed point and a periodic attractor as follows.

Definition 3. The state $x(t)$ is called a *fixed point* if $x(t+1) = x(t)$ holds. A set of states $\{x(t), x(t+1), \dots, x(t+p-1)\}$ ($p \geq 2$) is called a periodic attractor with period p if $x(t) = x(t+p-1)$ and $x(t) \neq x(t+q)$, $q = 1, 2, \dots, p-2$ hold.

A fixed point is also called a singleton attractor. In this thesis, we focus on a fixed point as one of the properties in the steady state.

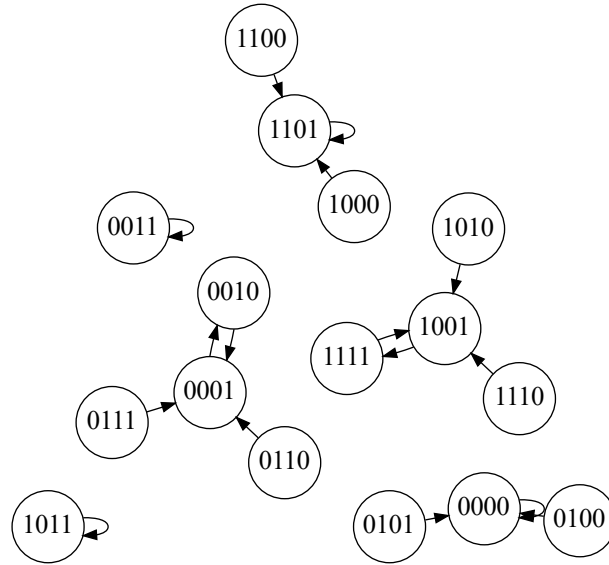


Figure 2.2: Transition graph in Example 1.

2.3 Boolean Networks with control inputs

Consider the following BN that is added the control input to the BN (2.1):

$$\begin{cases} x_1(t+1) = f_1(x(t), u(t)), \\ x_2(t+1) = f_2(x(t), u(t)), \\ \vdots \\ x_n(t+1) = f_n(x(t), u(t)), \end{cases} \quad (2.3)$$

where $u = [u_1, u_2, \dots, u_m] \in \{0, 1\}^m$ is the control input and f_i is a given Boolean function. We assume that for each element of the control input, any binary value can be set at each discrete time.

Example 4. Consider the following BN model for Th-lymphocyte differentiation [59]:

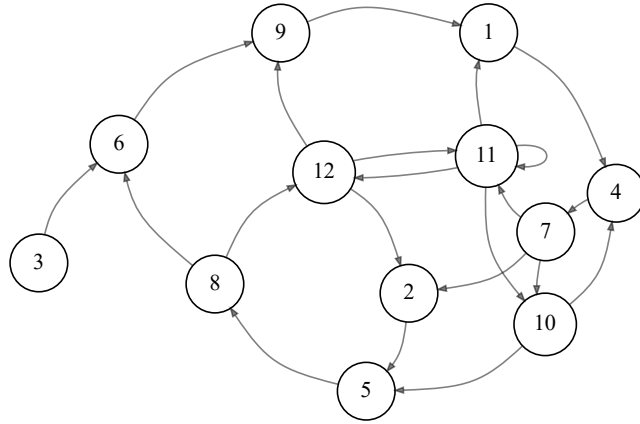


Figure 2.3: Interaction graph in Example 4.

$$\left\{ \begin{array}{l} x_1(t+1) = x_9(t) \vee x_{11}(t) \\ x_2(t+1) = \neg x_7(t) \wedge \neg x_{12}(t), \\ x_3(t+1) = 0, \\ x_4(t+1) = x_1(t) \wedge \neg x_{10}(t) \\ x_5(t+1) = x_2(t) \wedge \neg x_{10}(t) \\ x_6(t+1) = x_3(t) \wedge \neg x_8(t) \\ x_7(t+1) = x_4(t) \\ x_8(t+1) = x_5(t) \\ x_9(t+1) = x_6(t) \wedge \neg x_{12}(t) \\ x_{10}(t+1) = x_7(t) \vee x_{11}(t) \\ x_{11}(t+1) = \{x_7(t) \wedge \neg x_{12}(t) \vee x_{11}(t) \wedge \neg x_{12}(t)\} \oplus u_1(t) \\ x_{12}(t+1) = \{x_8(t) \wedge \neg x_{11}(t)\} \oplus u_2(t) \end{array} \right. \quad (2.4)$$

where

- x_1, x_2, x_3 are the concentration level of the lymphokines (IFN- γ , IL-4R, IL-12),
- x_4, x_5, x_6 are the concentration level of the receptors (IFN- γ R, IL-4R, IL-12R),
- x_7, x_8, x_9, x_{10} are the concentration level of the signaling transduction factors (STAT1, STAT6, STAT4, SOCS1),
- x_{11}, x_{12} are the concentration level of the transcription regulatory factors (T-bet, GATA-3).

Here, x_{11}, x_{12} are configured to receive control input u_1, u_2 . These control inputs have the effect of inverting the state, with a 0 resulting in no change and a 1 resulting in an inversion. The interaction graph and transition graph of the BN in Example 4 when control input 0 is always applied is shown in Fig. 2.3, Fig. 2.4.

This network has 3 fixed points and 4 periodic attractors:

$$\begin{aligned}
& [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], \quad [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0], \quad [0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1], \\
& \{[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1], [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]\}, \quad \{[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1], \\
& [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]\}, \\
& \{[0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1], \\
& [0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]\}, \quad \{[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], \\
& [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\}.
\end{aligned}$$

Here, we consider a control problem in which the fixed point $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]$ is the target state and all initial states must converge to the target state in a finite time. This network converges when u is given by the following feedback controller. Only this time, we omit the logical AND operator.

$$\begin{aligned}
u_1(t+1) &= g_1(t) = \neg x_{12}(t) \neg x_{11}(t) \neg x_1(t) \neg x_9(t) \neg x_8(t) \neg x_7(t) \neg x_6(t) \neg x_5(t) \neg x_4(t) \neg x_3(t) \neg x_2(t) \vee \\
&\quad x_2(t) x_8(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_{10}(t) \neg x_9(t) \neg x_7(t) \neg x_6(t) \neg x_5(t) \neg x_4(t) \neg x_1(t) \vee \\
&\quad x_{10}(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_9(t) \neg x_8(t) \neg x_7(t) \neg x_6(t) \neg x_5(t) \neg x_4(t) \neg x_3(t) \vee \\
&\quad x_5(t) x_8(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_2(t) \neg x_9(t) \neg x_7(t) \neg x_6(t) \neg x_4(t) \neg x_1(t) \vee \\
&\quad x_2(t) x_5(t) x_8(t) x_{12}(t) \neg x_{11}(t) \neg x_{10}(t) \neg x_9(t) \neg x_7(t) \neg x_4(t) \neg x_1(t) \vee \\
&\quad x_5(t) x_8(t) x_{10}(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_9(t) \neg x_7(t) \neg x_6(t) \neg x_4(t), \\
u_2(t+1) &= g_2(t) = x_2(t) x_5(t) x_8(t) x_{12}(t) \neg x_{11}(t) \neg x_{10}(t) \neg x_9(t) \neg x_7(t) \neg x_4(t) \neg x_1(t) \vee \\
&\quad x_8(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_9(t) \neg x_7(t) \neg x_6(t) \neg x_5(t) \neg x_4(t) \neg x_1(t) \vee \\
&\quad x_8(t) x_{10}(t) \neg x_{12}(t) \neg x_{11}(t) \neg x_9(t) \neg x_7(t) \neg x_6(t) \neg x_5(t) \neg x_4(t).
\end{aligned} \tag{2.5}$$

The transition graph of the BN in Example 4 that applied controller 2.5 is shown in Fig. 2.5.

2.4 Binary Decision Diagrams

A BDD is a data structure that efficiently represents a Boolean function. A BDD consists of two terminal nodes labeled with 0 and 1, and nodes labeled with variable names. Each node except for the terminal nodes has two child nodes, and one child node is chosen according to the value of the (parent) node. When the terminal node is reached by continuing to choose child nodes according to each value of node, the value of terminal node is the output of the function. It is known that a BDD is uniquely determined from a Boolean function when the order of variables on the graph is fixed. The data size of a BDD is highly sensitive to the order of variables. The problem of finding the optimal order of variables is known to be NP-complete[60]. Approximation algorithms of finding it have been studied (see, e.g., [61, 62, 63]).

The Apply operation has been proposed as a useful algorithm for handling logical equations on BDDs [49]. Apply operations can perform logical operators (AND, OR, XOR, etc.) between BDDs. The computation time of the Apply operation is roughly proportional to the total amount of input and output data. Fig. 2.6 shows the procedure to generate the logical expression $a \wedge b \vee \neg c$ by Apply operation.

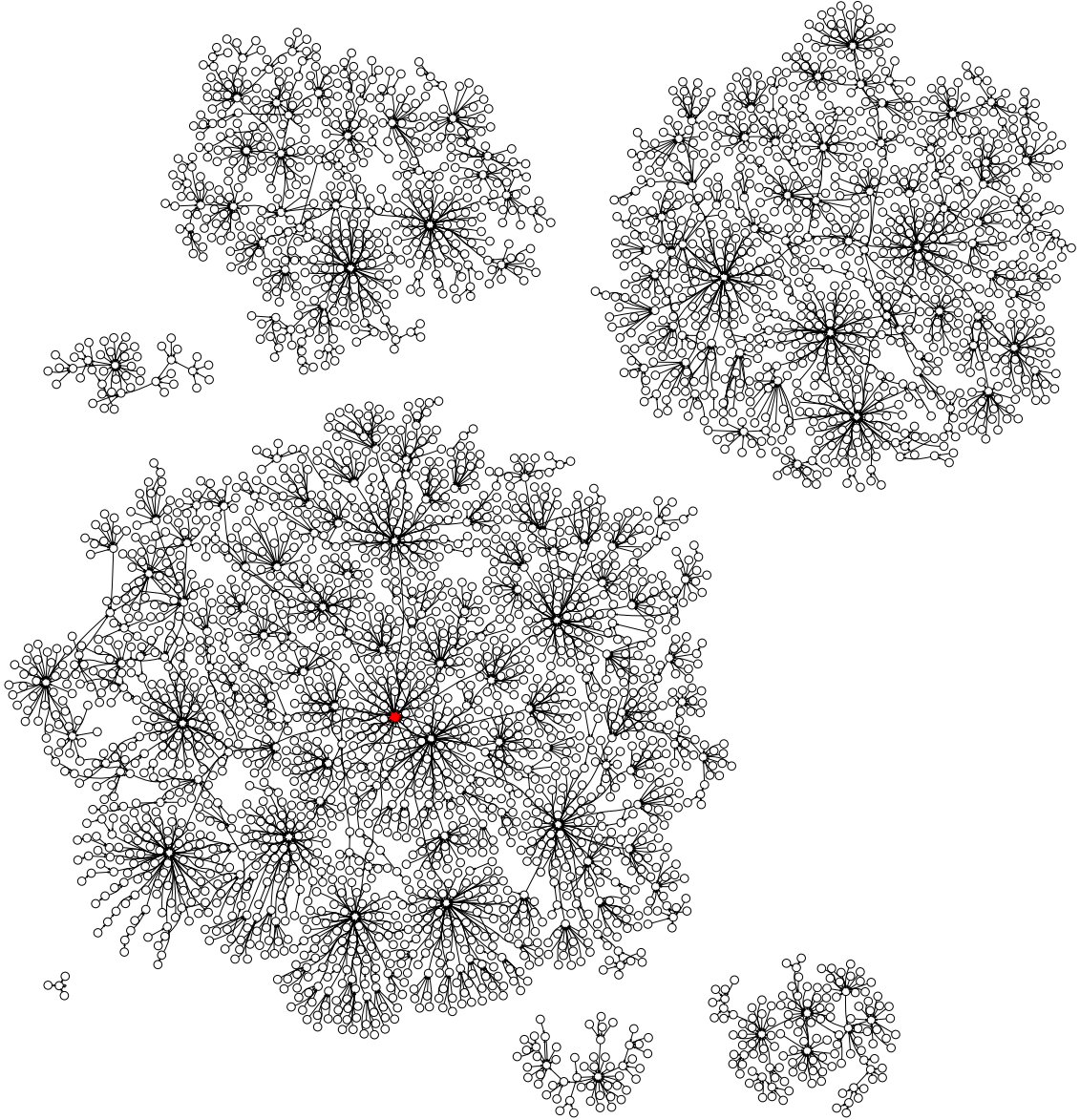


Figure 2.4: Transition graph in Example 2.4.

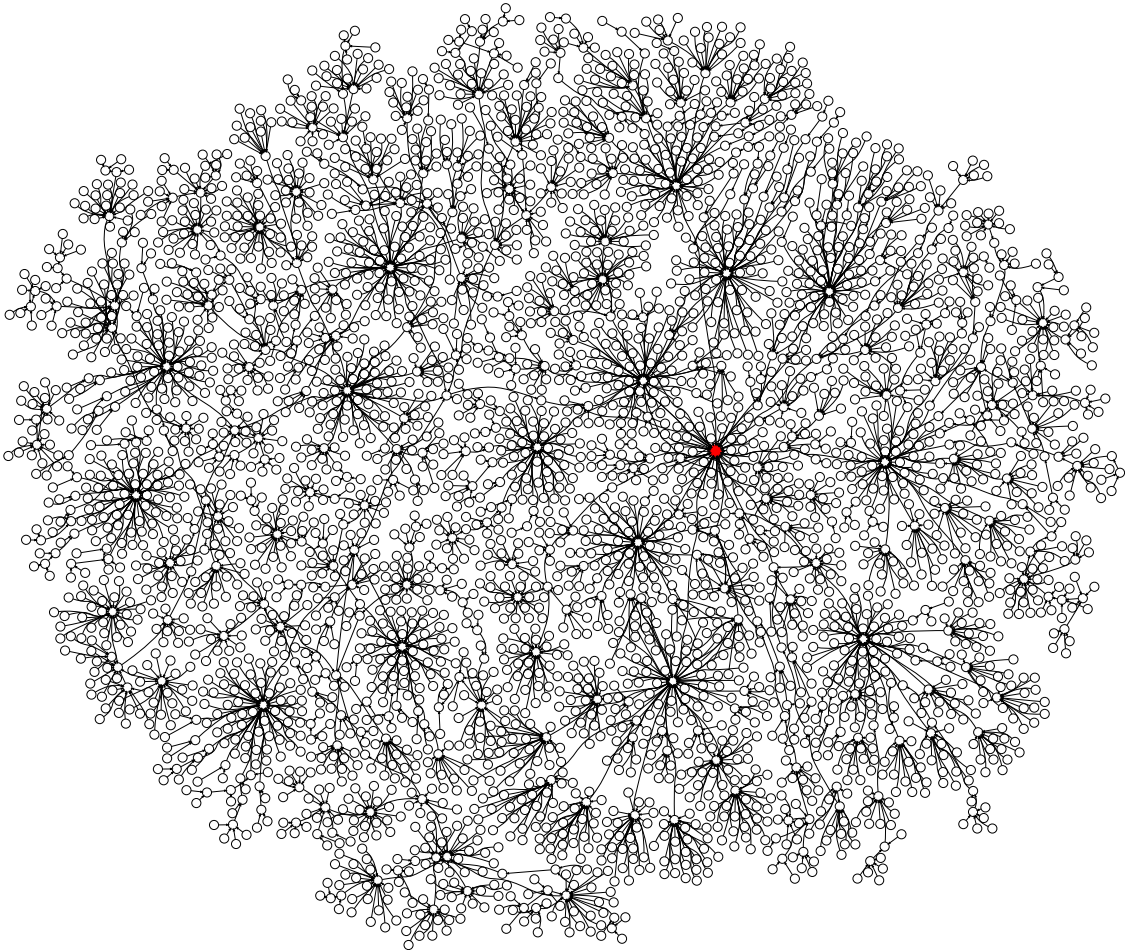


Figure 2.5: Transition graph in Example 2.4 that applied controller 2.5.

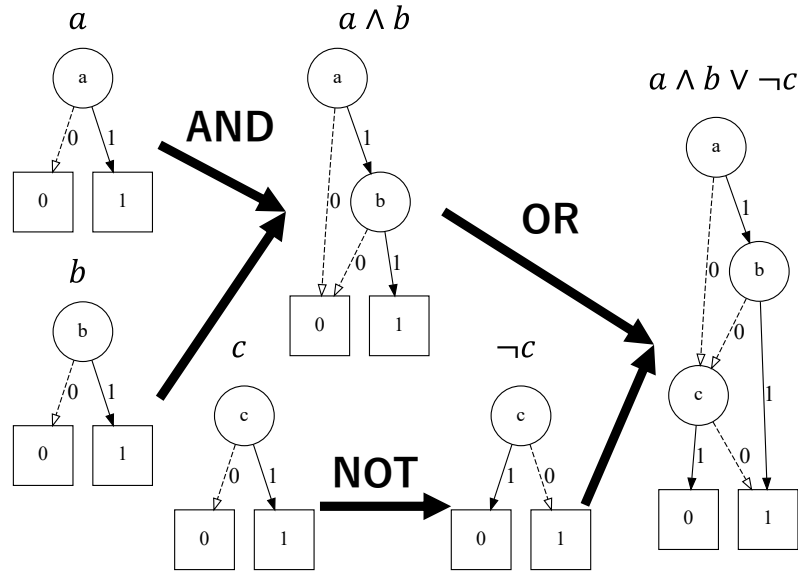


Figure 2.6: The procedure to generate the logical expression $a \wedge b \vee \neg c$ by Apply operation.

2.5 FVS-Based Model Reduction of Boolean Networks

2.5.1 Overview

In this section, we explain the model reduction method of BNs in which a fixed point is focused on. See [45, 47] for further details. In this method, we regard some of the state equations of a given BN as an algebraic equation. The state $x_i(t)$ is eliminated by substituting $f_i(\cdot)$ into $x_i(t)$. As a result, the number of states can be reduced.

First, we introduce the concept of a feedback vertex set (FVS). An FVS and a minimum FVS are defined as follows (see, e.g., [64, 65]).

Definition 5. A set of vertices of an interaction graph is called an FVS if removal of vertices results an acyclic graph. In particular, a feedback vertex set is called a minimum FVS if the number of its elements is minimum.

The computational complexity of finding a minimum FVS is NP-complete [65]. An approximate algorithm of finding it has been studied (see, e.g., [64]). When we utilize the model reduction method in [45], the number of states in the reduced BN is characterized by the number of vertices in the minimum FVS of the interaction graph. In addition, each fixed point in the reduced BN is a one-to-one correspondence with that in the original BN. To reduce a given BN, we need to eliminate states corresponding to vertices which are not included in the minimum FVS of the interaction graph.

Hereafter, the function f_i and the variable x_i corresponding to the minimum FVS of the interaction graph are denoted as $\hat{f}_i$ and $\hat{x}_i$, respectively. Under the above preparation, the procedure of the reduction method is given as follows.

Procedure of model reduction of a given BN:

Step 1: Find a minimum FVS $\mathcal{V}_F \subseteq \mathcal{V}$ of the interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Step 2: For $x_j(t+1) = f_j(\cdot)$, $j \in \mathcal{V} \setminus \mathcal{V}_F$, replace $x_j(t+1)$ with $x_j(t)$.

Step 3: Substitute $f_j(\cdot)$ into the variable $x_j(t)$ of the function $\hat{f}_i(\cdot)$, $i \in \mathcal{V}_F$. This operation is repeated until $x_j(t)$ is no longer in $\hat{f}_i(\cdot)$.

Step 4: Simplify $\hat{f}_i(\cdot)$.

In the case where finding a minimum FVS is difficult, we may replace it with an FVS with as few vertices as possible.

We present a simple biological example to explain this procedure.

Example 6. Consider the following BN model for the gene WNT5A[66]:

$$\begin{cases} x_1(t+1) = \neg x_6(t), \\ x_2(t+1) = \{\neg x_2(t) \wedge x_4(t) \wedge x_6(t)\} \\ \quad \vee \{x_2(t) \wedge x_4(t)\} \vee \{x_2(t) \wedge x_6(t)\}, \\ x_3(t+1) = \neg x_7(t), \\ x_4(t+1) = x_4(t), \\ x_5(t+1) = x_2(t) \vee \neg x_7(t), \\ x_6(t+1) = x_3(t) \vee x_4(t), \\ x_7(t+1) = \neg x_2(t) \vee x_7(t). \end{cases}$$

where

- x_1 is the concentration level of the gene WNT5A,
- x_2 is the concentration level of the gene pirin,
- x_3 is the concentration level of the gene S100P,
- x_4 is the concentration level of the gene RET1,
- x_5 is the concentration level of the gene MART1,
- x_6 is the concentration level of the gene HADHB,
- x_7 is the concentration level of the gene STC2.

The interaction graph of this BN is shown in Fig. 2.7.

In Step 1, the minimum FVS of this BN is given by

$$\mathcal{V}_F = \{2, 4, 7\}.$$

In Step 2, $x_j(t+1)$, $j \in \mathcal{V} \setminus \mathcal{V}_F = \{1, 3, 5, 6\}$ is replaced with $x_j(t)$. In the beginning of Step 3, x_i and $f_i(\cdot)$, $i \in \mathcal{V}_F$ are replaced with $\hat{x}_i$ and $\hat{f}_i(\cdot)$, respectively. Then, we can

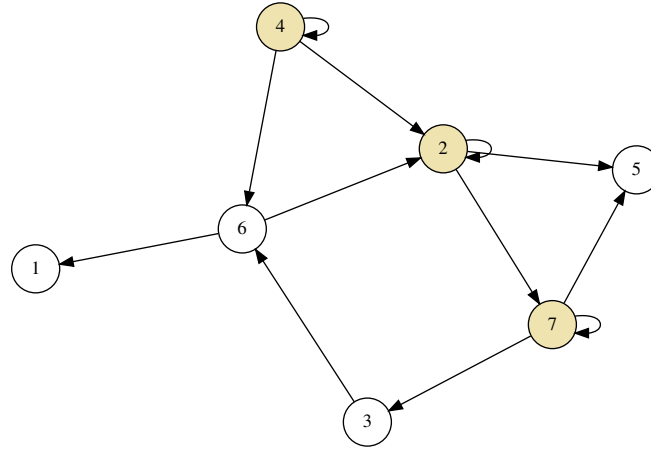


Figure 2.7: Interaction graph in Example 6, where the highlighted vertices belong to the minimum FVS.

obtain

$$\begin{cases} x_1(t) &= \neg x_6(t), \\ \hat{x}_2(t+1) &= \{\neg \hat{x}_2(t) \wedge \hat{x}_4(t) \wedge x_6(t)\} \vee \{\hat{x}_2(t) \wedge \hat{x}_4(t)\} \vee \{\hat{x}_2(t) \wedge x_6(t)\}, \\ x_3(t) &= \neg \hat{x}_7(t), \\ \hat{x}_4(t+1) &= \hat{x}_4(t), \\ x_5(t) &= \hat{x}_2(t) \vee \neg \hat{x}_7(t), \\ x_6(t) &= x_3(t) \vee \hat{x}_4(t), \\ \hat{x}_7(t+1) &= \neg \hat{x}_2(t) \vee \hat{x}_7(t), \end{cases}$$

In Step 3, the obtained algebraic equations are substituted into the state equations of $\hat{x}_2$. Then, we can obtain

$$\begin{aligned} \hat{x}_2(t+1) &= \{\neg \hat{x}_2(t) \wedge \hat{x}_4(t) \wedge x_6(t)\} \vee \{\hat{x}_2(t) \wedge \hat{x}_4(t)\} \vee \{\hat{x}_2(t) \wedge x_6(t)\} \\ &= \{\neg \hat{x}_2(t) \wedge \hat{x}_4(t) \wedge (x_3(t) \vee \hat{x}_4(t))\} \vee \\ &\quad \{\hat{x}_2(t) \wedge \hat{x}_4(t)\} \vee \{\hat{x}_2(t) \wedge (x_3(t) \vee \hat{x}_4(t))\} \\ &= \{\neg \hat{x}_2(t) \wedge \hat{x}_4(t) \wedge (\neg \hat{x}_7(t) \vee \hat{x}_4(t))\} \vee \\ &\quad \{\hat{x}_2(t) \wedge \hat{x}_4(t)\} \vee \{\hat{x}_2(t) \wedge (\neg \hat{x}_7(t) \vee \hat{x}_4(t))\} \end{aligned}$$

In Step 4, by simplifying the above state equation, we obtain the following reduced BN:

$$\begin{cases} \hat{x}_2(t+1) = \{\hat{x}_2(t) \wedge \neg \hat{x}_7(t)\} \vee \hat{x}_4(t), \\ \hat{x}_4(t+1) = \hat{x}_4(t), \\ \hat{x}_7(t+1) = \neg \hat{x}_2(t) \vee \hat{x}_7(t). \end{cases}$$

We discuss fixed points. The original BN has four fixed points $([0, 1, 0, 1, 1, 1, 1]^\top, [0, 1, 1, 0, 1, 1, 0]^\top, [0, 1, 1, 1, 1, 1, 0]^\top, [1, 0, 0, 0, 0, 0, 1]^\top)$. The reduced BN has also four fixed points $([1, 1, 1]^\top, [1, 0, 0]^\top, [1, 1, 0]^\top, [0, 0, 1]^\top)$. From these fixed points, we see that

each fixed point in the original BN is a one-to-one correspondence with that in the reduced BN. In this example, fixed points can be distinguishable by using three states.

Finally, we present one example of the control methods using the reduced BN. Consider the following problem:

Problem 1. Assume that one state except for x_2 can be replaced with a constant control input, which can be arbitrarily set to 0 or 1. Find a constant control input such that x_2 converges to 0.

Such input selection problem is important in control of BNs [67, 68, 45]. The solution for this problem is that x_4 is replaced with a constant control input, and $x_4(t) = 1$ for any k . This can be obtained from the reduced BN, because if $\hat{x}_4(t) = 1$ holds, then $\hat{x}_2(t+1) = 1$. Of course, this solution can be obtained from also the original BN. However, since x_6 is eliminated from the reduced BN, it is easy to find the solution. Thus, the reduced BN can be used to the control problem focusing on the fixed point. $\square$

In the model reduction method considered here, only fixed points are preserved. The dynamical behavior is changed, and is not preserved (in other words, the dynamical behavior is ignored). Applications of a reduced BN are limited. On the other hand, the state of any BN necessarily converges to either fixed points or periodic attractors (there is no notion of divergence). Hence, even if only information on fixed points is preserved, then there is a possibility that the reduced BN can be applied to several control problems focusing on fixed points. The above example of the control problems is one of the applications of the reduced BN.

2.5.2 Model Reduction Using the Semi-Tensor Product Method

Using a simple example, we briefly explain model reduction of BNs using the STP method. Here, we consider the model reduction method in [45].

Consider the BN model in Example 1, where the minimum FVS is given by $\mathcal{V}_F = \{1, 3\}$. Using the procedure of model reduction in Section 2.5, we can obtain the reduced BN model as follows:

$$\begin{cases} \hat{x}_1(t+1) = \hat{x}_1(t), \\ \hat{x}_3(t+1) = \hat{x}_3(t). \end{cases} \quad (2.6)$$

Next, some notations related to the STP method are defined. For two matrices $A \in \mathcal{R}^{m \times n}$ and $B \in \mathcal{R}^{p \times q}$, the STP of A and B is defined by $A \ltimes B = (A \otimes I_{c/n})(B \otimes I_{c/p})$, where $\otimes$ is the Kronecker product, and c is the least common multiple of n and p . Let δ_n^i denote the i -th column of I_n . Let $\Delta_n = \{\delta_n^i, i = 1, 2, \dots, n\}$ denote the set of δ_n^i . The matrix $A \in \{0, 1\}^{m \times n}$ is called a logical matrix if each column of A is given by some element of Δ_m . The matrix $[\delta_n^{i_1}, \delta_n^{i_2}, \dots, \delta_n^{i_s}]$ is simply denoted by $\delta_n[i_1, i_2, \dots, i_s]$. Finally, for any $x \in \{0, 1\}$, we define $\bar{x} := [x, 1-x]^\top$. See, e.g., [40] for further details.

Using the STP method, the BN model (2.2) can be expressed by

$$\begin{cases} \bar{x}_1(t+1) = \bar{x}_1(t), \\ \bar{x}_2(t+1) = L_2 \ltimes \bar{x}_1(t) \ltimes \bar{x}_3(t), \\ \bar{x}_3(t+1) = L_3 \ltimes \bar{x}_2(t) \ltimes \bar{x}_4(t), \\ \bar{x}_4(t+1) = L_4 \ltimes \bar{x}_1(t) \ltimes \bar{x}_3(t), \end{cases}$$

where $L_2 = \delta_2[2, 1, 2, 2]$, $L_3 = \delta_2[2, 2, 1, 2]$, and $L_4 = \delta_2[1, 1, 1, 2]$. According to the procedure of model reduction, we can obtain

$$\hat{x}_3(t+1) = L_3 \ltimes (L_2 \ltimes \hat{x}_1(t) \ltimes \hat{x}_3(t)) \ltimes (L_4 \ltimes \hat{x}_1(t) \ltimes \hat{x}_3(t)).$$

From the property of the STP, this equation can be rewritten as

$$\begin{aligned}\bar{\hat{x}}_3(t+1) &= L_3 \ltimes (L_2 \otimes L_4) \\ &\quad \ltimes ((\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)) \ltimes (\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t))).\end{aligned}$$

Furthermore, the following relation holds:

$$\begin{aligned}(\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)) \ltimes (\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)) \\ = \delta_{16}[1, 6, 11, 16] \ltimes (\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)).\end{aligned}$$

Using this relation, we can obtain

$$\begin{aligned}\bar{\hat{x}}_3(t+1) &= L_3 \ltimes (L_2 \otimes L_4) \ltimes \delta_{16}[1, 6, 11, 16] \ltimes (\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)) \\ &= \delta_2[1, 2, 1, 2] \ltimes (\bar{\hat{x}}_1(t) \ltimes \bar{\hat{x}}_3(t)) \\ &= \bar{\hat{x}}_3(t).\end{aligned}$$

Hence, we can obtain the reduced BN model as follows:

$$\begin{cases} \bar{\hat{x}}_1(t+1) = \bar{\hat{x}}_1(t), \\ \bar{\hat{x}}_3(t+1) = \bar{\hat{x}}_3(t), \end{cases}$$

which is equivalent to (2.6). Thus, the STP method can be utilized in model reduction. However, the size of matrices grows explosively. In this simple example, the size of $L_2 \otimes L_4$ is 4×16 , which depends on $|\mathcal{N}_i|$. We conclude that the implementation with the STP method is not appropriate for large-scale BNs.

Chapter 3

BDD-Based Implementation of Model Reduction

3.1 Introduction

We consider implementing the model reduction method that the information on singleton attractors is preserved [45]. In gene regulatory networks, singleton attractors represent cell types or states of cells [69], and are important to decide characteristics of cells. The dynamical behavior of the original BN is not preserved. However, the reduced model obtained by the method in [45] can be used in design of constant control inputs (see Example 6 of Section 2.5). In control of gene regulatory networks, constant control inputs are frequently used (see, e.g., [70, 58]). Hence, the model reduction method that the singleton attractors are preserved is important. In the method of [45], some Boolean functions are substituted into other one. By performing substitutions repeatedly, the number of states is reduced to the number of vertices in a feedback vertex set (FVS) in the interaction graph in which interactions between states are represented (see Definition 5 for the definition of FVSs). Then, the calculation order on BDDs must be characterized. In this chapter, we propose a method for deriving it based on the interaction graph. Using the obtained calculation order, the reduced model can be derived by only the apply operations on BDDs

Furthermore, to show the effectiveness of the proposed method, numerical examples of large-scale BNs that the STP method cannot be used are presented. Through numerical examples, we show that for such large-scale BNs, the reduced model can be obtained fast by using the proposed method.

This chapter is organized as follow. In Section 3.2, an implementation method using BDDs is proposed. In Section 3.3, numerical examples of large-scale BNs are presented. In Section 3.4, we conclude this chapter.

3.2 BDD-Based Implementation Method

Consider implementing the model reduction method in the previous section. In the last decade, the STP method has been widely used in analysis and control of BNs (see, e.g., [22, 40]). In the STP method, the Boolean function f_i in (2.1) is represented by the matrix with the size of $2 \times 2^{|\mathcal{N}_i|}$. When we substitute some Boolean functions into a certain Boolean

function, matrices with a larger size must be manipulated (see Section 2.5.2 for further details). Hence, it is not appropriate to use the STP method for large-scale BNs.

In this chapter, using a BDD, we propose a new method for implementing the model reduction of BNs. In the proposed method, the reduced BN is automatically generated from Boolean functions, an interaction graph, and its minimum FVS. First, we explain the outline of BDDs and their technical issue in the model reduction method of BNs. Next, we propose a new implementation method of model reduction using BDDs.

3.2.1 Technical Issue of BDDs in Model Reduction

A BDD is a data structure used to represent Boolean functions (see Section 3.2). The biggest advantage of using BDDs as a representation of Boolean functions is that the Apply operations can be executed in a calculation time that is almost proportional to the total amount of input/output data.

However, the Apply operations cannot be directly used to the model reduction method of BNs. The Apply operations are not suitable for substituting some Boolean function into some variable (state) of the other Boolean function. Because in order to perform substitutions, the BDD must be generally transformed into a Boolean function.

3.2.2 Proposed Method

We consider overcoming the above technical issue by deriving an appropriate calculation order of the Apply operations. We present a very simple example on the operation of substitution.

Example 7. Consider substituting $b = c \wedge \neg d$ into $a \wedge b \vee \neg c$, where $a, b, c \in \{0, 1\}$. Let $B(f)$ denote the BDD of the Boolean function f . Substituting $B(b) = B(c \wedge \neg d)$ into $B(a \wedge b \vee \neg c)$ cannot be directly performed. According to the key idea of the proposed method, first, we derive $B(c \wedge \neg d)$. Using it, we can derive $B(a \wedge (c \wedge \neg d) \vee \neg c) = B(a) \wedge B(c \wedge \neg d) \vee B(\neg c)$. Thus, the operation of substitution can be performed by using the appropriate calculation order. $\square$

For a simple Boolean function, the calculation order is obvious. However, in more complicated cases, such as Example 6, finding the calculation order is not easy. Because there exist multiple Boolean functions that must be used in substitutions. In the proposed method, the calculation order is derived based on the interaction graph and its minimum FVS. Boolean functions of a given BN are not required. Using the appropriate calculation order, we can implement the model reduction method with only BDDs.

Consider one Boolean function $\hat{f}_i(\cdot)$, $i \in \mathcal{V}_F$, where no assumption is made for $\hat{f}_i(\cdot)$ (in other words, the proposed method can be applicable to any BN). Suppose that the minimum FVS $\mathcal{V}_F$ is obtained from the interaction graph $\mathcal{G}$. We may use an FVS with as few vertices as possible. Let $\mathcal{M}_i$ denote the vertex set obtained by deleting the searched vertices from $\mathcal{N}_i$. In the initial step, $\mathcal{M}_i$ is set as $\mathcal{M}_i := \mathcal{N}_i$. Let c_o denote the vector representing the calculation order in ascending order. Let s_t denote the vector representing the vertices that the parent vertices must be searched. In the proposed procedure, the dimensions of c_o and s_t are changed. The case where c_o and s_t are scalars is also allowed. The vector s_t implies a stack. In the initial step, c_o and s_t are empty vectors. The proposed procedure is presented as follows.

Procedure to reduce the Boolean function $\widehat{f}_i(\cdot)$:

Step 1: Set $s_t := i$, and go to Step 2.

Step 2: If s_t is empty, then go to Step 5, otherwise go to Step 3.

Step 3: Let j denote the last element in s_t . If $\mathcal{M}_j = \emptyset$ holds, then delete j from s_t , set $c_o := [c_o, j]$, and go to Step 2. Otherwise (i.e., $\mathcal{M}_j \neq \emptyset$ holds), choose arbitrarily the vertex k from $\mathcal{M}_j$, and go to Step 4.

Step 4: Delete the vertex k from $\mathcal{M}_j$. If $k \notin \mathcal{V}_F$ holds and k is not included in elements of c_o , then set $s_t := [s_t, k]$. Go to Step 3.

Step 5: Perform the Apply operation to $\widehat{f}_i(\cdot)$ according to the calculation order give by c_o .

See also the flowchart shown in Fig. 3.1. In the above procedure, $s_t := i$ means that s_t is updated to i . In addition, $s_t := [s_t, k]$ means that s_t is updated to $[s_t, k]$, that is, the dimension of s_t increases.

To explain this procedure, we present a simple example.

Example 8. Consider finding the calculation order to reduce $\widehat{f}_2(\cdot)$ in Example 6. The minimum FVS is given by $\mathcal{V}_F = \{2, 4, 7\}$. Here, the loop between Step 2 and Step 4 is simply called the loop. The values of the variables after each loop are shown in Table 3.1.

In Step 1, s_t is set as $s_t = 2$.

1st loop: In Step 2, since s_t is not empty, go to Step 3. In Step 3, j is set as $j = 2$. Since $\mathcal{M}_2 = \mathcal{N}_2 = \{2, 4, 6\} \neq \emptyset$ holds, k is chosen as $k = 6$ and go to Step 4. In Step 4, $\mathcal{M}_2 = \{2, 4, 6\}$ is updated to $\mathcal{M}_2 = \{2, 4\}$. Since $k = 6 \notin \mathcal{V}_F$ and k is not included in elements of c_o , the vector s_t is updated to $s_t = [2, 6]$. Go to Step 3.

2nd loop: In Step 3, j is set as $j = 6$. Since $\mathcal{M}_6 = \mathcal{N}_6 = \{3, 4\} \neq \emptyset$ holds, k is chosen as $k = 3$ and go to Step 4. In Step 4, $\mathcal{M}_6 = \{3, 4\}$ is updated to $\mathcal{M}_6 = \{4\}$. Since $k = 3 \notin \mathcal{V}_F$ and k is not included in elements of c_o , the vector s_t is updated to $s_t = [2, 6, 3]$. Go to Step 3.

3rd loop: In Step 3, j is set as $j = 3$. Since $\mathcal{M}_3 = \{7\} \neq \emptyset$ holds, k is chosen as $k = 7$ and go to Step 4. In Step 4, $\mathcal{M}_3 = \{7\}$ is updated to $\mathcal{M}_3 = \emptyset$. Since $k = 7 \in \mathcal{V}_F$ holds, the vector s_t is not updated. Go to Step 3.

4th loop: In Step 3, j is set as $j = 3$ again. Since $\mathcal{M}_3 = \emptyset$ holds, the vector s_t is updated to $s_t = [2, 6]$. The vector c_o is also updated to $c_o = 3$. Go to Step 2.

5th loop: In Step 2, since s_t is not empty, go to Step 3. In Step 3, j is set as $j = 6$. Since $\mathcal{M}_6 = \{4\} \neq \emptyset$ holds, k is chosen as $k = 4$ and go to Step 4. In Step 4, $\mathcal{M}_6 = \{4\}$ is updated to $\mathcal{M}_6 = \emptyset$. Since $k = 4 \in \mathcal{V}_F$ holds, the vector s_t is not updated. Go to Step 3.

6th loop: In Step 3, j is set as $j = 6$ again. Since $\mathcal{M}_6 = \emptyset$ holds, the vector s_t is updated to $s_t = 2$. The vector c_o is also updated to $c_o = [3, 6]$. Go to Step 2.

7th loop: In Step 2, since s_t is not empty, go to Step 3. In Step 3, j is set as $j = 2$. Since $\mathcal{M}_2 = \{2, 4\} \neq \emptyset$ holds, k is chosen as $k = 2$ and go to Step 4. In Step 4, $\mathcal{M}_2 = \{2, 4\}$ is updated to $\mathcal{M}_2 = \{4\}$. Since $k = 2 \in \mathcal{V}_F$ holds, the vector s_t is not updated. Go to Step 3.

8th loop: In Step 3, j is set as $j = 2$ again. Since $\mathcal{M}_2 = \{4\} \neq \emptyset$ holds, k is chosen as $k = 4$ and go to Step 4. In Step 4, $\mathcal{M}_2 = \{4\}$ is updated to $\mathcal{M}_2 = \emptyset$. Since $k = 4 \in \mathcal{V}_F$ holds, the vector s_t is not updated. Go to Step 3.

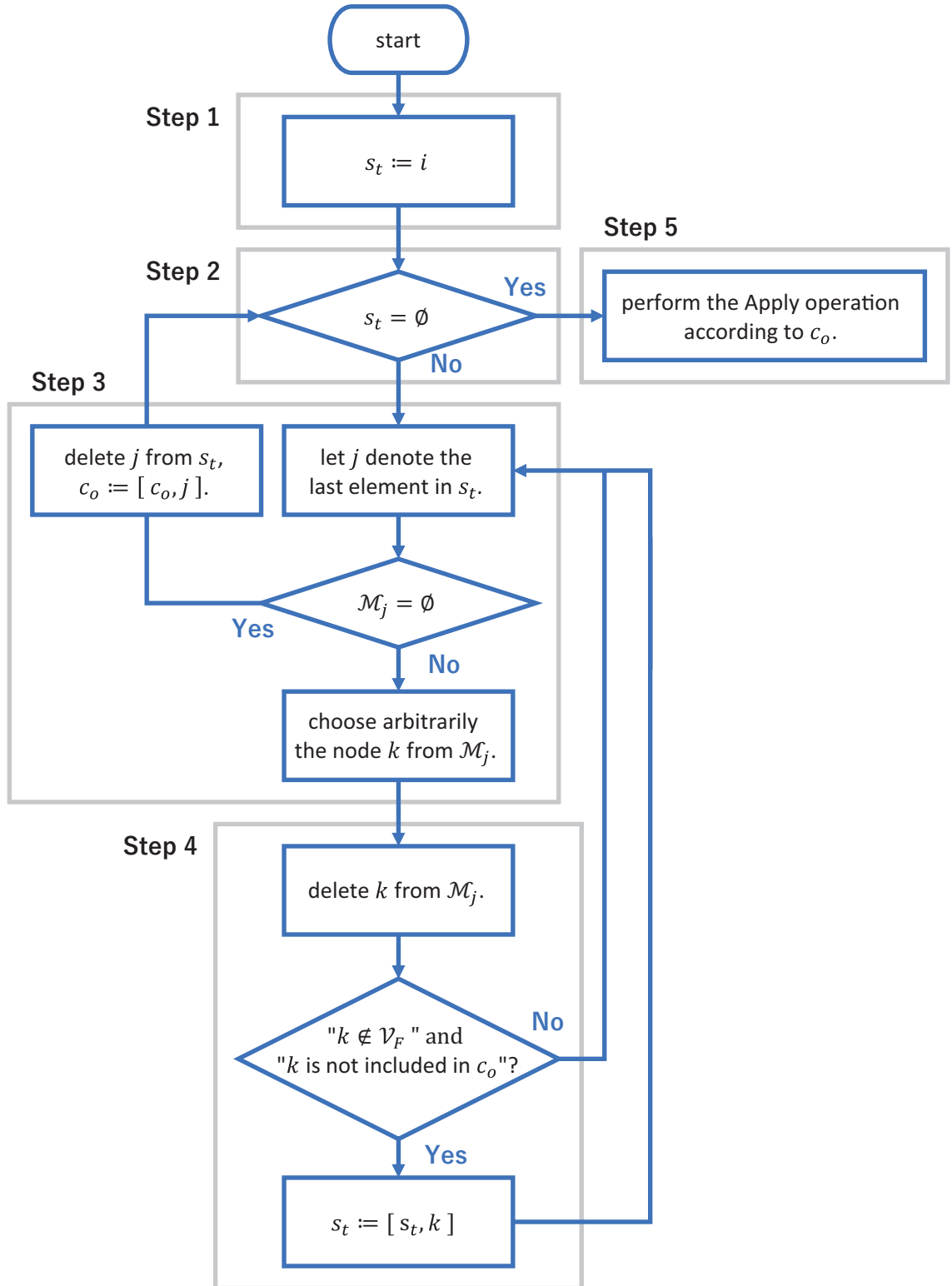
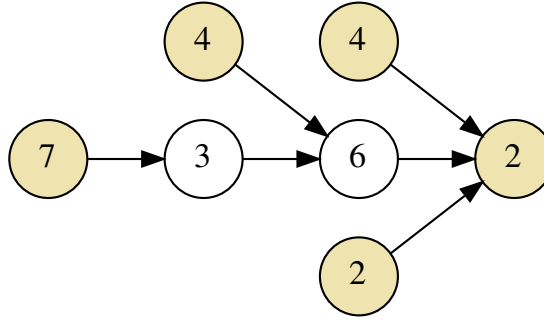


Figure 3.1: Flowchart of the proposed procedure.

Table 3.1: Value of the variable after each loop

loop	c_0	s_t	$\mathcal{M}_i$
0	$\emptyset$	2	$\mathcal{M}_2 = \{2, 4, 6\}$
1	$\emptyset$	2, 6	$\mathcal{M}_2 = \{2, 4\}, \mathcal{M}_6 := \{3, 4\}$
2	$\emptyset$	2, 6, 3	$\mathcal{M}_2 = \{2, 4\}, \mathcal{M}_6 := \{4\}, \mathcal{M}_3 := \{7\}$
3	$\emptyset$	2, 6, 3	$\mathcal{M}_2 = \{2, 4\}, \mathcal{M}_6 := \{4\}, \mathcal{M}_3 := \emptyset$
4	3	2, 6	$\mathcal{M}_2 = \{2, 4\}, \mathcal{M}_6 := \{4\}$
5	3	2, 6	$\mathcal{M}_2 = \{2, 4\}, \mathcal{M}_6 := \emptyset$
6	3, 6	2	$\mathcal{M}_2 = \{2, 4\}$
7	3, 6	2	$\mathcal{M}_2 = \{4\}$
8	3, 6	2	$\mathcal{M}_2 = \emptyset$
9	3, 6, 2	$\emptyset$	

Figure 3.2: Parent vertices connected to the vertices in $c_o = [3, 6, 2]$, where the vertex set $\{2, 4, 7\}$ is the minimum FVS of the interaction graph.

9th loop: In Step 3, j is set as $j = 2$ again. Since $\mathcal{M}_2 = \emptyset$ holds, s_t is updated to an empty vector. The vector c_o is also updated to $c_o = [3, 6, 2]$. Go to Step 2.

Finally, in Step 2, since s_t is empty, go to Step 5. In Step 5, we reduce $\hat{f}_2(\cdot)$ according to $c_o = [3, 6, 2]$ (see also Fig. 3.2 and Fig. 3.3). First, $f_3 = \neg \hat{x}_7(t)$ is substituted into f_6 . Next, the obtained $f_6 = \hat{x}_7(t) \vee \hat{x}_4(t)$ is substituted into $\hat{f}_2$. Thus, we can obtain

$$\hat{f}_2(\cdot) = \{\hat{x}_2(t) \wedge \neg \hat{x}_7(t)\} \vee \hat{x}_4(t).$$

□

Finally, we consider the termination of the proposed procedure. For the interaction graph and some vertex $i \in \mathcal{V}_F$, some notations are defined. Let α denote the length of the longest path among the paths from i to vertices included in $\mathcal{V}_F \setminus \{i\}$. Let β denote the number of the paths from i to vertices included in $\mathcal{V}_F \setminus \{i\}$. Let γ_j denote the length of the path from i to $j \in \mathcal{V}_F \setminus \{i\}$. We define $\gamma := \sum_j (\gamma_j - 1)$.

As an example, consider the interaction graph shown in Fig. 2.7. We set $i = 2$. We can enumerate the following four paths: $2 \rightarrow 2$, $2 \rightarrow 4$, $2 \rightarrow 6 \rightarrow 3 \rightarrow 7$, and $2 \rightarrow 6 \rightarrow 4$. The parameters α , β , and γ can be derived as $\alpha = 3$, $\beta = 4$, and $\gamma = 2$, respectively.

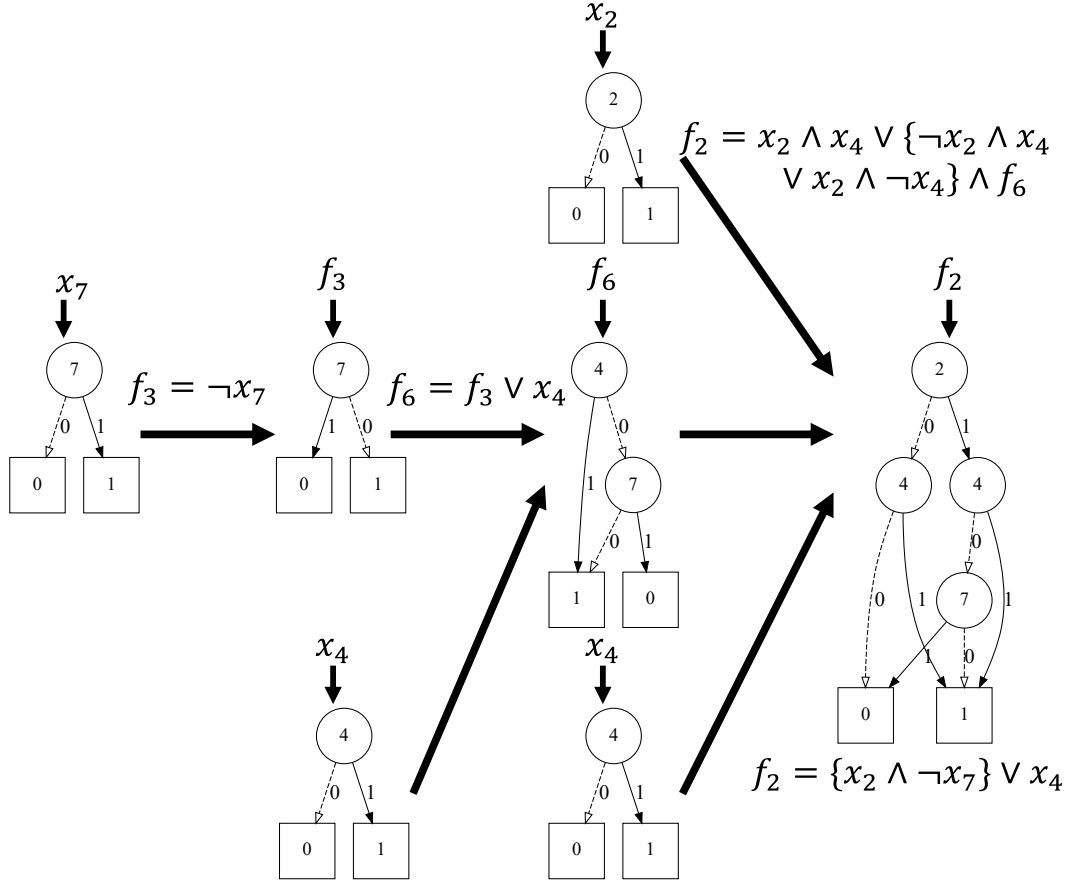


Figure 3.3: Reduction of $\hat{f}_2(\cdot)$ based on Fig. 3.2, where $\hat{\cdot}$ is omitted.

The parameter γ represents the number of the vertices not included in $\mathcal{V}_F$ in these paths (duplicated vertices are not eliminated).

We can obtain the following result on the termination of the proposed procedure.

Theorem 9. *In the procedure to reduce the Boolean function $\hat{f}_i(\cdot)$, the number of times that the loop between Step 2 and Step 4 is iterated is at most $2\alpha + \beta + \gamma - 1$, that is, this procedure is terminated.*

Proof. In the loop between Step 2 and Step 4, the cases where the loop occurs are the following four cases (see Fig. 3.1): (i) $\mathcal{M}_j = \emptyset$, (ii) $k \in \mathcal{V}_F$, (iii) k is included in c_o , and (iv) $s_t := [s_t, k]$. First, the length of s_t is given by α . This is because the length of s_t is not changed in the case of $k \in \mathcal{V}_F$. Hence, the number of times that Case (i) occurs is given by α . Next, the number of times that Case (ii) occurs is given by β . Third, the elements of c_o except for the last element are given by the labels of vertices not included in $\mathcal{V}_F$. Hence, the number of times that Case (iii) occurs is given by at most γ . Finally, since s_t is set as $s_t := i$ in Step 1, the number of times that Case (iv) occurs is given by $\alpha - 1$. Thus, the number of times that the loop is iterated is at most $\alpha + \beta + \gamma + (\alpha - 1) = 2\alpha + \beta + \gamma - 1$. $\square$

In Example 8, the number of times that Case (iii) occurs is zero. Hence, the number of times that the loop is iterated is given by $2\alpha + \beta - 1 = 9$.

3.3 Numerical Examples

In this section, we present numerical examples of large-scale BNs to validate the proposed method.

Example 10. Consider the BN with 60 states, where the interaction graph is randomly generated such that the scale-free property is satisfied for the in-degrees of the vertices, and logical operations are randomly assigned. The interaction graph is shown in Fig. 3.4. One of the FVSs for this interaction graph can be obtained as

$$\mathcal{V}_F = \{3, 11, 15, 17, 18, 22, 24, 26, 31, 34, 37, 41, 43, 51, 53, 57, 58, 60\}.$$

This FVS is calculated by a heuristic method, and its minimality is not guaranteed. Here, we focus on the Boolean function f_{17} given by

$$\begin{aligned} f_{17} = & x_{16}(t) \vee x_{19}(t) \vee x_{21}(t) \vee x_{27}(t) \vee x_{45}(t) \vee x_{47}(t) \vee x_{55}(t) \vee \neg x_{43}(t) \vee \neg x_{59}(t) \\ & \vee \{x_6(t) \wedge x_{58}(t)\} \vee \{x_{14}(t) \wedge x_{53}(t)\} \vee \{x_{26}(t) \wedge x_{51}(t)\} \vee \{x_{10}(t) \wedge \neg x_{15}(t)\} \\ & \vee \{x_{12}(t) \wedge x_{23}(t) \wedge \neg x_{54}(t)\} \vee \{x_{11}(t) \wedge \neg x_{37}(t) \wedge \neg x_{39}(t) \wedge \neg x_{44}(t)\} \\ & \vee \{x_3(t) \wedge x_{20}(t) \wedge \neg x_{29}(t) \wedge x_{32}(t) \wedge x_{36}(t)\} \\ & \vee \{x_4(t) \wedge x_9(t) \wedge x_{13}(t) \wedge x_{17}(t) \wedge x_{35}(t) \wedge x_{41}(t) \wedge x_{46}(t) \wedge x_{50}(t)\}, \end{aligned}$$

where f_{17} has 37 states (variables). When we consider the model reduction using the STP method (see 2.5.2), we must need to manipulate the matrix with the size 2×2^{37} . However, this matrix cannot be constructed by using the standard environment of Python and MATLAB. For such large-scale BN, we can utilize the proposed procedure.

Using the proposed procedure, we can obtain the calculation order c_o as follows:

$$\begin{aligned} c_o = & [55, 4, 46, 19, 25, 10, 39, 44, 52, 20, 45, 21, 6, 9, 29, 42, 50, 14, 30, 47, 35, \\ & 1, 32, 27, 13, 36, 56, 33, 49, 7, 28, 23, 12, 16, 59, 54, 17]. \end{aligned}$$

According to the obtained c_o , the operation of substitution is performed. Fig. 3.5 shows the parent vertices connected to the vertices in c_o . Based on c_o , we can obtain the reduced Boolean function $\hat{f}_{17}$ as follows:

$$\hat{f}_{17} = \hat{x}_3 \vee \hat{x}_{11} \vee \neg \hat{x}_{15} \vee \neg \hat{x}_{18} \vee \neg \hat{x}_{26} \vee \hat{x}_{37} \vee \hat{x}_{41} \vee \neg \hat{x}_{43} \vee \hat{x}_{51} \vee \neg \hat{x}_{60}.$$

Applying the proposed procedure to the vertices included in $\mathcal{V}_F$, we can obtain the interaction graph shown in Fig. 3.6, which corresponds to the reduced BN.

Example 11. We present the computation time of the proposed procedure. Consider 50 Boolean networks with 100 states, which are randomly generated such that the scale-free property is satisfied for the in-degrees of the vertices, and logical operations are randomly assigned.

We present the computation result using the proposed procedure. In 50 reduced BNs, the number of the states is derived as follows:

- The average number: 23.4,
- The maximum number: 29,
- The minimum number: 14.

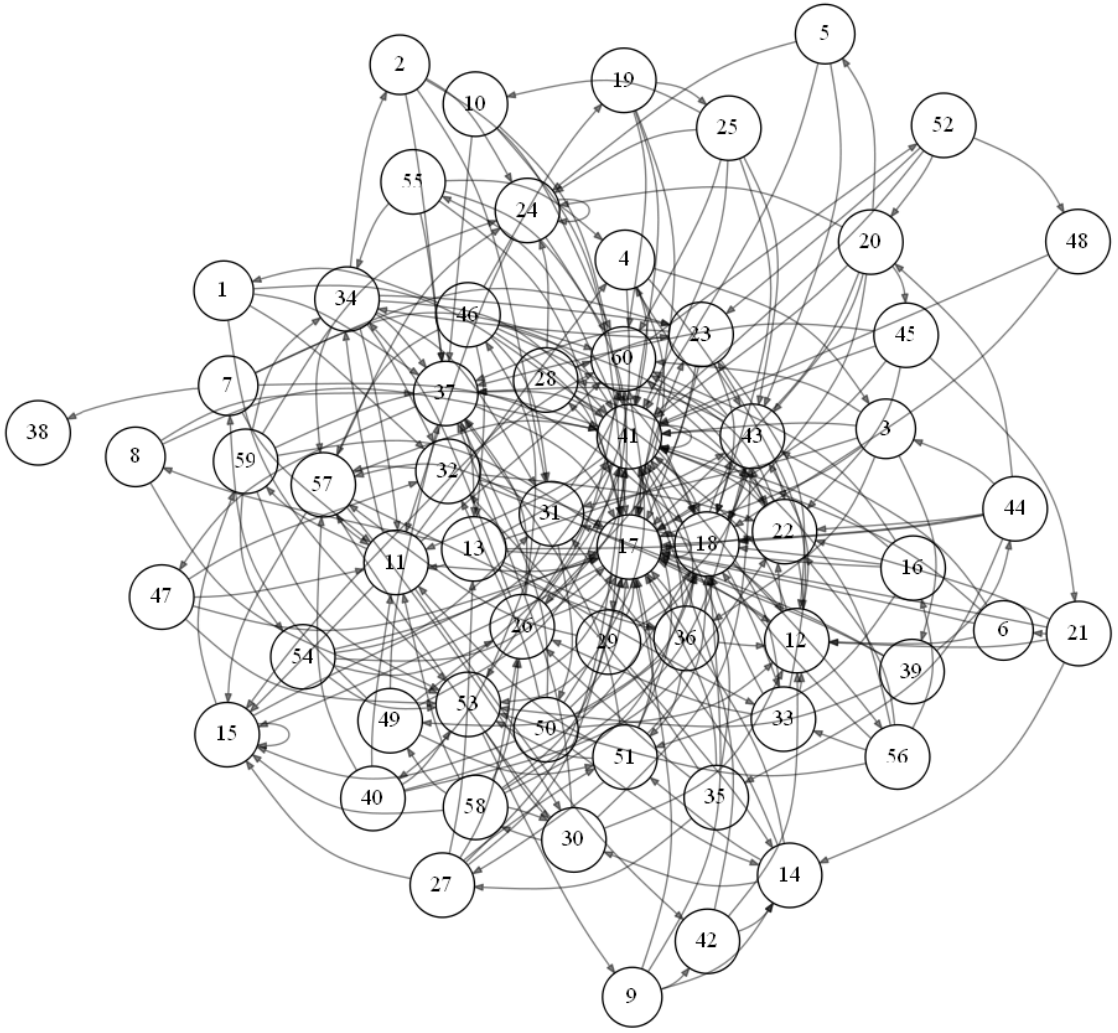


Figure 3.4: Interaction graph in Example 10.

From these values, we see that the number of the states can be widely reduced using the proposed procedure.

For 50 cases, the computation time of the proposed procedure is derived as follows:

- The average computation time: 2.76 sec,
- The worst computation time: 10.99 sec,
- The best computation time: 0.37 sec,

where we use Python 3.8.5 on the computer (CPU: Intel Core i7-4770K 3.50GHz, Memory: 8GB, OS: Windows 10). From these values, we see that the proposed procedure can be fast performed for large-scale BNs.

3.4 Conclusion

In this chapter, we proposed a model reduction method of BNs using BDDs. In order to utilize BDDs efficiently, a method for deriving the calculation order was proposed based on the interaction graph of a given BN. The effectiveness of the proposed method was presented by numerical examples on large-scale BNs that the STP method cannot be applied.

In the proposed method, a given BN is reduced to a BN with the number of elements of an FVS as the number of states. However, the minimality of the number of states is not discussed. The number of elements of an FVS is generally not minimal [71]. It is one of the future efforts to develop a method for deriving a further reduced model. In this chapter, we focus on fixed points. It is also one of the future efforts to extend the proposed method to the case of periodic attractors (see Definition 3 for the definition of periodic attractors). Furthermore, it is important to develop a method for evaluating the gap between the original BN and the reduced BN and a model reduction method considering the dynamical behavior based on the gap. In Example 6 of Section 2.5, we presented a simple example of control methods using the reduced BN model. Further development of control methods using the reduced BN model is also important.

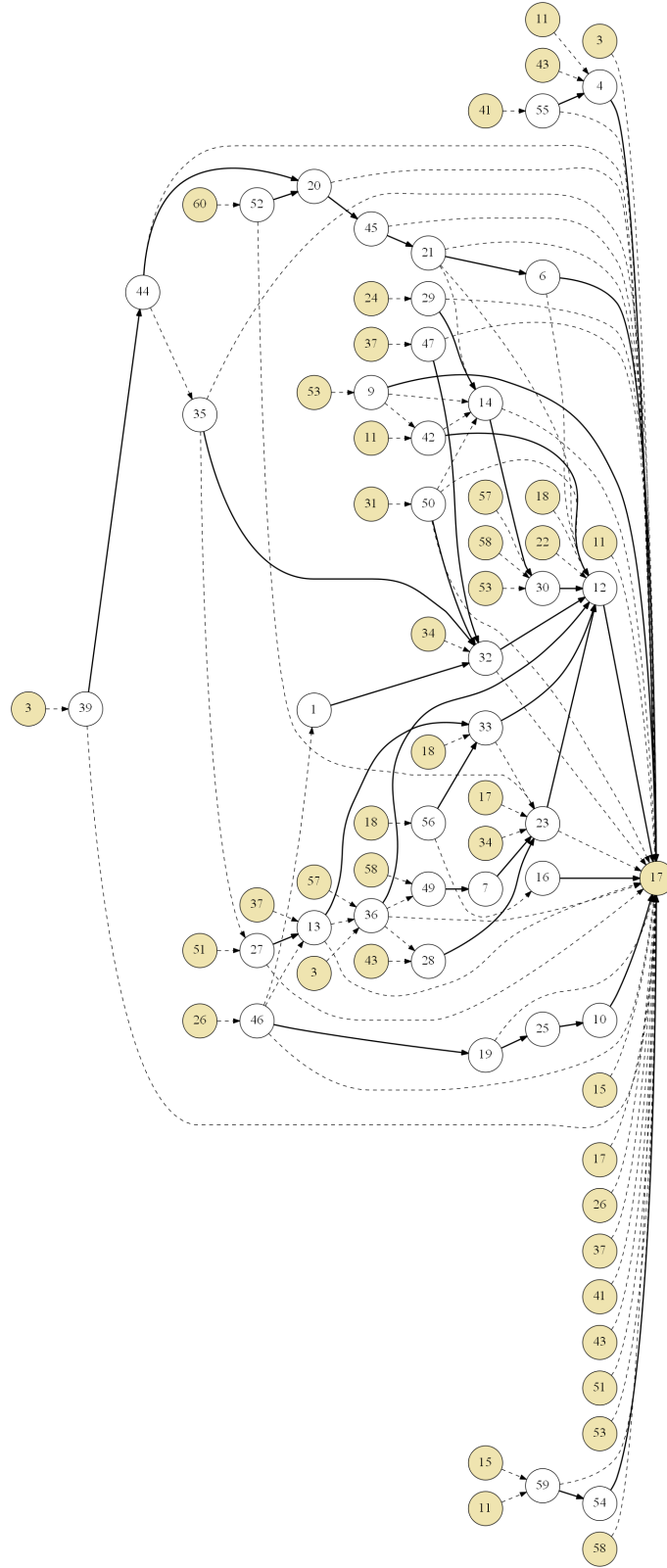


Figure 3.5: Parent vertices connected to the vertices in c_o , where the dotted line implies that the vertex is not added to s_t in Step 4 of the procedure.

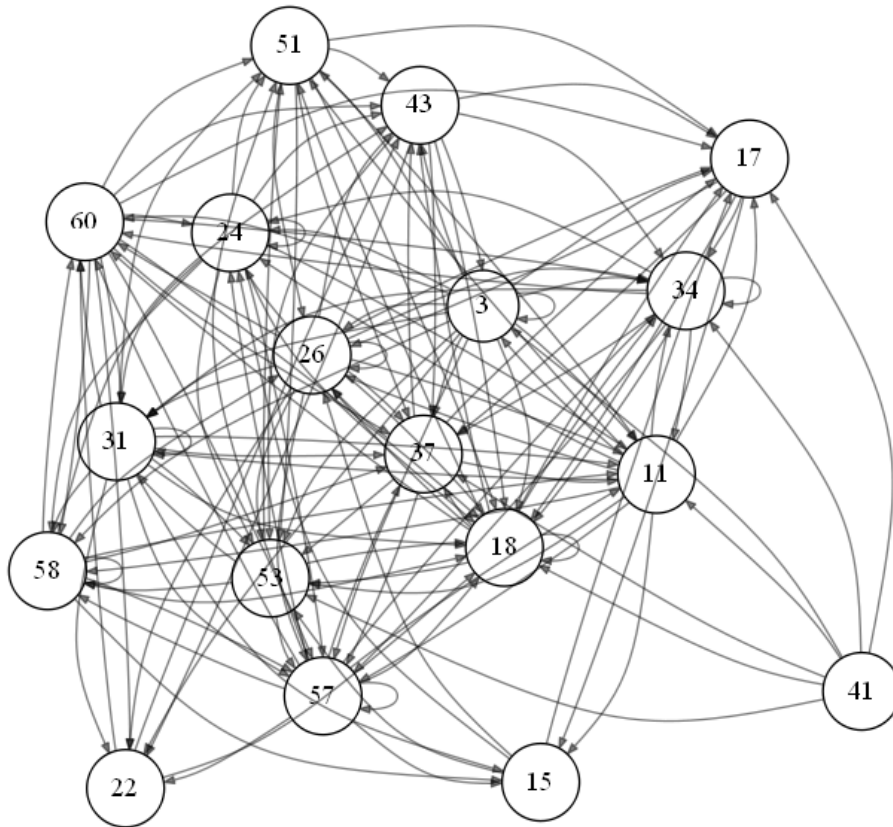


Figure 3.6: Interaction graph for the reduced BN in Example 10.

Chapter 4

Model Reduction Based on Complement and Absorption Laws

4.1 Introduction

In Chapter 3, the model reduction methods based on [45, 47] that the information on fixed points (singleton attractors) is preserved have been proposed. In [45, 50, 47], a feedback vertex set (FVS), which is a subset of the vertex set of the graph expressing interactions between elements of the state, plays an important role (see Section 2.5 for details of an FVS). This method can reduce the dimension of the state of a given BN to the number of vertices of an FVS. However, the number of vertices of an FVS is in general not a minimal dimension of the state of the reduced model (see, e.g., [71]). The reduced BN obtained by using the methods in [45, 47] may be possible to be further reduced.

In this chapter, a new method for model reduction of BNs is proposed. In the proposed method, the interaction graph obtained from a given BN is used. The interaction graph represents interactions between elements of the state in a given BN (logical operations are ignored). The reduced BN obtained by using the methods in [45, 47] may be reduced by substitution operations through paths in the interaction graph. In the proposed method, using the interaction graph, we find substitution operations such that complement and absorption laws of Boolean functions can be applied to a Boolean function. As a result, reduction of states is achieved.

This chapter is organized as follows. In Section 4.2, a model reduction method using the interaction graph is proposed. In Section 4.3, a numerical example is presented to demonstrate the proposed method. In Section 4.4, we conclude this chapter.

4.2 Proposed Method

Using the FVS-based model reduction method, the dimension of the state is reduced to the number of vertices in the FVS of a given interaction graph. In general, the number of vertices in the minimum FVS is not a minimal number of the reduced states that fixed points are distinguishable. In this section, we propose a method for further reducing states based on a given interaction graph.

Hereafter, for simplicity of notations, $\hat{\cdot}$ is omitted in $\hat{f}_i$ and $\hat{x}_i$.

4.2.1 Motivating Example

We present a motivating example.

Example 12. Consider the following BN with the three-dimensional state:

$$\begin{cases} f_1 = x_1, \\ f_2 = \neg x_1 \wedge x_2, \\ f_3 = x_1 \wedge x_2 \wedge x_3. \end{cases}$$

All vertices of the interaction graph of this BN have a self-loop. Then, a minimum FVS is given by $\{1, 2, 3\}$. That is, the dimension of the state cannot be reduced by the FVS-based model reduction method. Here, we focus on the Boolean function f_3 . Consider substituting $x_2 = f_2$ into f_3 . We can obtain

$$\begin{aligned} f'_3 &= f_3|_{x_2=f_2} \\ &= x_1 \wedge x_2 \wedge x_3|_{x_2=f_2} \\ &= x_1 \wedge (\neg x_1 \wedge x_2) \wedge x_3 \\ &= (x_1 \wedge \neg x_1) \wedge x_2 \wedge x_3 \\ &= 0 \wedge x_2 \wedge x_3 \\ &= 0. \end{aligned}$$

Hence, we can obtain $x_3(t+1) = 0$, which implies that x_3 can be reduced. Finally, we discuss fixed points. The original BN has three fixed points $([0, 0, 0], [0, 1, 0], [1, 0, 0])$. The reduced BN has also three fixed points $([0, 0], [0, 1], [1, 0])$. Even if the above substitution operation is performed, each fixed point in the original BN is a one-to-one correspondence with that in the reduced BN. $\square$

From this example, we see that there is a possibility that further reduction can be achieved by substitution operations. Since substitution operations are only continued ignoring the information about FVSs, in a similar to the FVS-based method, each fixed point in the original BN is a one-to-one correspondence with that in the reduced BN. In the proposed method, we focus on the following complement and absorption laws for Boolean functions:

$$A \wedge \bar{A} = 0, \quad A \vee \bar{A} = 1, \tag{4.1}$$

$$A \vee (A \wedge B) = A, \quad A \wedge (A \vee B) = A, \tag{4.2}$$

where $A, B \in \{0, 1\}$. We also focus on the property that the i -th element x_i of the state satisfying either $x_i(t+1) = 0$ or $x_i(t+1) = 1$ can be reduced. Moreover, the FVS-based model reduction method may be applied to the BN obtained by performing further substitution operations and deleting constant Boolean functions. This is because the interaction graph is changed by these operations.

4.2.2 Proposed Model Reduction Method

First, we define a walk and a path for a directed graph. A walk is defined as a sequence of vertices $[v_0, \dots, v_T]$ that the directed edges (v_i, v_{i+1}) , $0 \leq i \leq T-1$ exist in the directed

graph. A path is defined as a walk where all vertices are different. If there exists a path $[v_0, v_1, \dots, v_{T-1}, v_T]$ on the interaction graph, we can perform the following substitution calculation:

$$\begin{aligned} f'_{v_1} &= f_{v_1}, \\ f'_{v_2} &= f_{v_2}|_{x_{v_1}=f'_{v_1}}, \\ f'_{v_2} &= f_{v_2}|_{x_{v_1}=f'_{v_1}, x_{v_2}=f'_{v_2}}, \\ &\vdots \\ f'_{v_T} &= f_{v_T}|_{x_{v_1}=f'_{v_1}, x_{v_2}=f'_{v_2}, \dots, x_{v_{T-1}}=f'_{v_{T-1}}}. \end{aligned} \quad (4.3)$$

We remark that f'_{v_T} has the variable x_{v_0} .

In the proposed method, we consider performing (4.1) or (4.2) by substituting $x_l = f_l(\cdot)$, $l \in \mathcal{N}_i$ into the function $f_i(\cdot)$. A necessary condition for performing (4.1) or (4.2) is that $f_i(\cdot)$ and $f_l(\cdot)$ have at least one common variable (i.e., there exists $j \in \mathcal{N}_i \cap \mathcal{N}_l$). This variable corresponds to A in (4.1) and (4.2). In the case where there exists no $j \in \mathcal{N}_i \cap \mathcal{N}_l$, if there exists a path $[j, v_0, v_1, \dots, v_{K-1}, v_K, i]$, and $j \in \mathcal{N}_i$ holds, then it may be possible to perform (4.1) or (4.2) by the following substitution operations:

$$\begin{aligned} f'_{v_0} &= f_{v_0}, \\ f'_{v_1} &= f_{v_1}|_{x_{v_0}=f'_{v_0}}, \\ &\vdots \\ f'_{v_K} &= f_{v_K}|_{x_{v_0}=f'_{v_0}, \dots, x_{v_{K-1}}=f'_{v_{K-1}}}, \\ f'_i &= f_i|_{x_{v_0}=f'_{v_0}, \dots, x_{v_{K-1}}=f'_{v_{K-1}}, x_{v_K}=f'_{v_K}}. \end{aligned}$$

In summary, if there exists a path $p = [v_0, \dots, v_T]$ that contains the start point in the parent vertices of the end point ($v_0 \in \mathcal{N}_{v_T}$), it may be possible to reduce f_T by the substitution calculation along the path.

Next, we propose a procedure of model reduction using the absorption and complement laws. In the proposed procedure, for each nodes i on the interaction graph, the paths satisfying $[j, \dots, i]$, $j \in \mathcal{N}_i$ are listed in order from the shortest, and the path that reduce the function f_i can be found from these paths. Let $\mathcal{F}$ denote the set of indexed Boolean functions. Let $\mathcal{I}_{\mathcal{F}}$ denote the index set of $\mathcal{F}$.

Procedure of model reduction using the absorption and complement laws:

Step 1: Apply the FVS-based model reduction method to each Boolean function. Substitute constant functions into other Boolean functions. Set $\mathcal{F} = \emptyset$ (i.e., $\mathcal{I}_{\mathcal{F}} = \emptyset$).

Step 2: Enumerate paths of length 1 in the interaction graph (self-loops are excluded). The set of enumerated paths is defined by $\mathcal{P} := \{p_i \mid i \in \mathcal{A} \subset \{1, 2, \dots\}\}$.

Step 3: For each $p_i = [v_{i0}, \dots, v_{iT}]$, find the path set $\{[v_j, v_{i0}, \dots, v_{iT}] \mid \forall j \in \mathcal{N}_{v_{i0}}\}$. Update the path set $\mathcal{P} := \cup_{i \in \mathcal{A}} \{[v_j, v_{i0}, \dots, v_{iT}] \mid \forall j \in \mathcal{N}_{v_{i0}}\}$, which is denoted by $\mathcal{P} = \{p_i \mid i \in \mathcal{A}\}$.

Step 4: For each $p_i = [v_{i0}, \dots, v_{iT}] \in \mathcal{P}$, if $v_{i0} \in \mathcal{N}_{v_{iT}}$ and $\mathcal{I}_{\mathcal{F}} \cap \{v_{i0}, \dots, v_{iT}\} = \emptyset$ are satisfied, obtain $f'_{v_{iT}}$ by performing the substitution operations according to p_i such as

(4.3). If the number of states in $f'_{v_{iT}}$ is smaller than that in the original $f_{v_{iT}}$, then update $\mathcal{F}$ as $\mathcal{F} := \mathcal{F} \cup \{f'_{v_{iT}}\}$.

Step 5: For each i of $\mathcal{P} = \{p_i \mid i \in \mathcal{A}\}$, delete all p_i that satisfy $\mathcal{I}_{\mathcal{F}} \cap \{v_{i0}, \dots, v_{iT}\} \neq \emptyset$ from the set $\mathcal{P}$.

Step 6: Follow the following conditional branch.

- **Case 1:** If $\mathcal{P} \neq \emptyset$, then go to Step 3.
- **Case 2:** If $\mathcal{P} = \emptyset$ and $\mathcal{F} \neq \emptyset$, then update f_i , $i \in \mathcal{I}_{\mathcal{F}}$ in a given BN as $f_i := f'_i \in \mathcal{F}$. Go to Step 1.
- **Case 3:** If $\mathcal{P} = \emptyset$ and $\mathcal{F} = \emptyset$, then terminate the procedure.

We explain each step.

In Step 1, the FVS-based model reduction method is applied to a given BN. Substitution operations with constant functions are also performed.

In Step 2, paths of length 1 are enumerated for each vertex.

In Step 3, the length of each path is extended by one.

In Step 4, substitution operations using each path are performed. A Boolean function may be simplified by the complement law (4.1) and the absorption law (4.2), because substitution operations such that these laws occur are included. When a Boolean function is simplified by substitution operations, this Boolean function is added to the set $\mathcal{F}$.

In Step 5, paths related to paths used in simplification of Boolean functions are deleted.

In Case 1 of Step 6, the length of each path is extended by one. Because substitution operations using paths of a longer length are required. In Case 2 of Step 6, a given BN is reduced. Since the graph structure is changed, we perform Step 1 again. In Case 3 of Step 6, the procedure is terminated.

We present a simple example to demonstrate the proposed procedure.

Example 13. Consider the following 5-node BN:

$$\begin{cases} f_1 = x_1 \vee \neg x_4, \\ f_2 = (x_1 \wedge \neg x_5) \vee x_2 \vee (\neg x_3 \wedge \neg x_4), \\ f_3 = x_3 \vee x_4, \\ f_4 = x_1 \vee x_2 \vee x_4 \vee x_5, \\ f_5 = x_4 \vee x_5. \end{cases} \quad (4.4)$$

The interaction graph of this BN is shown in the Fig. 4.1. Since all vertices have a self-loop, that is, a minimum FVS is given by all vertices, this BN cannot be reduced by the FVS-based model reduction method.

In Step 1 of the proposed procedure, only the action $\mathcal{F} := \emptyset$ is required.

In Step 2, we enumerate all the paths of length 1 in the interaction graph as follows:

$$\mathcal{P} = \{ [4, 1], [1, 2], [3, 2], [4, 2], [5, 2], [4, 3], [1, 4], [2, 4], [5, 4], [4, 5] \}$$

(i.e., $\mathcal{A} = \{1, 2, \dots, 10\}$).

In Step 3, we extend the paths by one on the start point side. Then, we can obtain the following set $\mathcal{P}$:

$$\begin{aligned} \mathcal{P} = \{ & [2, 4, 1], [5, 4, 1], [4, 1, 2], [4, 3, 2], [1, 4, 2], [5, 4, 2], [4, 5, 2], \\ & [1, 4, 3], [2, 4, 3], [5, 4, 3], [1, 2, 4], [3, 2, 4], [5, 2, 4], [1, 4, 5], [2, 4, 5] \}. \end{aligned}$$

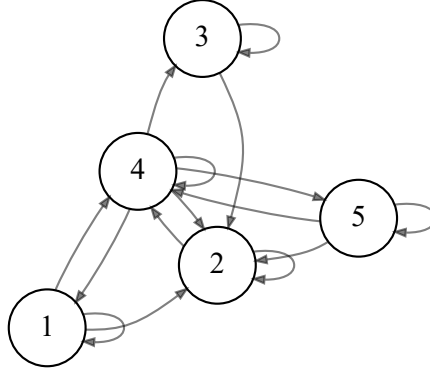


Figure 4.1: Interaction graph of the BN (4.4).

In Step 4, the paths whose parent vertices of the end point contain the start point are given as follows:

$$[4, 1, 2], [4, 3, 2], [1, 4, 2], [5, 4, 2], [4, 5, 2], [1, 2, 4], [5, 2, 4].$$

Substitution calculations according to these paths are performed, but there are no Boolean functions that can be reduced.

In Step 5, since $\mathcal{F} = \emptyset$ holds, any path is not deleted.

In Step 6, since the condition of Case 1 holds, go to Step 3.

In Step 3, we extend the paths by one on the start point side again.

$$P = \{ [3, 2, 4, 1], [5, 2, 4, 1], [5, 4, 1, 2], [1, 4, 3, 2], [5, 4, 3, 2], \\ [1, 4, 5, 2], [1, 2, 4, 3], [5, 2, 4, 3], [1, 2, 4, 5], [3, 2, 4, 5] \}$$

In Step 4, the paths whose parent vertices of the end point contains the start point are given as follows:

$$[5, 4, 1, 2], [1, 4, 3, 2], [5, 4, 3, 2], [1, 4, 5, 2].$$

We perform substitution calculations according to these paths. Then, by substitution operations using the $[5, 4, 1, 2]$, the Boolean function f_2 with five states can be reduced to

f'_2 with four states as follows.

$$\begin{aligned}
f'_4 &= x_1 \vee x_2 \vee x_4 \vee x_5, \\
f'_1 &= x_1 \vee \neg x_4|_{x_4=f'_4} \\
&= x_1 \vee \neg(x_1 \vee x_2 \vee x_4 \vee x_5) \\
&= x_1 \vee (\neg x_2 \wedge \neg x_4 \wedge \neg x_5), \\
f'_2 &= (x_1 \wedge \neg x_5) \vee x_2 \vee (\neg x_3 \wedge \neg x_4)|_{x_1=f'_1, x_4=f'_4} \\
&= ((x_1 \vee (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)) \wedge \neg x_5) \vee x_2 \vee (\neg x_3 \wedge \neg(x_1 \vee x_2 \vee x_4 \vee x_5)) \\
&= x_2 \vee (x_1 \wedge \neg x_5) \vee ((\neg x_2 \wedge \neg x_4 \wedge \neg x_5) \vee ((\neg x_2 \wedge \neg x_4 \wedge \neg x_5) \wedge (\neg x_1 \wedge \neg x_3))) \\
&= x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_2 \wedge \neg x_4 \wedge \neg x_5) \\
&= x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_4 \wedge \neg x_5),
\end{aligned}$$

where the absorption law (4.2) is used in calculation of f'_2 . The obtained Boolean function f'_2 is added to the $\mathcal{F}$.

In Step 5, we delete the paths through the vertex of the function of $\mathcal{F} = \{f'_2\}$ from the set $\mathcal{P}$, and we obtain $\mathcal{P} = \emptyset$.

In Step 6, since the condition of Case 2 holds, f_2 in a given BN (4.4) is replaced with $f'_2 \in \mathcal{F}$ as follows:

$$\begin{cases} f_1 = x_1 \vee \neg x_4, \\ f_2 = x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_4 \wedge \neg x_5), \\ f_3 = x_3 \vee x_4, \\ f_4 = x_1 \vee x_2 \vee x_4 \vee x_5, \\ f_5 = x_4 \vee x_5. \end{cases} \quad (4.5)$$

In Step 1, since all the nodes of the interaction graph of the modified BN (4.5) have a self-loop, only the action $\mathcal{F} := \emptyset$ is required.

In Step 2, we enumerate all the paths of length 1 in the interaction graph as follows:

$$\mathcal{P} = \{ [4, 1], [1, 2], [4, 2], [5, 2], [4, 3], [1, 4], [2, 4], [5, 4], [4, 5] \}.$$

In Step 3, we extend the path of $\mathcal{P}$ by one on the start point side as follows:

$$\begin{aligned}
\mathcal{P} = \{ & [2, 4, 1], [5, 4, 1], [4, 1, 2], [1, 4, 2], [5, 4, 2], [4, 5, 2], [1, 4, 3], \\ & [2, 4, 3], [5, 4, 3], [1, 2, 4], [5, 2, 4], [1, 4, 5], [2, 4, 5] \}.
\end{aligned}$$

In Step 4, the paths whose parent vertices of the end point contains the start point are given by

$$[4, 1, 2], [1, 4, 2], [5, 4, 2], [4, 5, 2], [1, 2, 4], [5, 2, 4].$$

We perform substitution calculations according to these paths. Then, by substitution operations using two paths $[4, 5, 2]$ and $[5, 2, 4]$, the Boolean functions f_2 and f_4 can be

reduced as follows:

$$\begin{aligned}
 f'_5 &= x_4 \vee x_5, \\
 f'_2 &= x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_4 \wedge \neg x_5)|_{x_5=f'_5} \\
 &= x_2 \vee (x_1 \wedge \neg(x_4 \vee x_5)) \vee (\neg x_4 \wedge \neg(x_4 \vee x_5)) \\
 &= x_2 \vee (\neg x_4 \wedge \neg x_5) \vee ((\neg x_4 \wedge \neg x_5) \wedge x_1) \\
 &= x_2 \vee (\neg x_4 \wedge \neg x_5)
 \end{aligned}$$

(the absorption law (4.2) is used) and

$$\begin{aligned}
 f'_2 &= x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_4 \wedge \neg x_5), \\
 f'_4 &= x_1 \vee x_2 \vee x_4 \vee x_5|_{x_2=f'_2} \\
 &= x_1 \vee (x_2 \vee (x_1 \wedge \neg x_5) \vee (\neg x_4 \wedge \neg x_5)) \vee x_4 \vee x_5 \\
 &= x_1 \vee x_2 \vee (x_1 \wedge \neg x_5) \vee \neg(x_4 \vee x_5) \vee (x_4 \vee x_5) \\
 &= 1
 \end{aligned}$$

(the complement law (4.1) is used). Two Boolean functions f'_2 and f'_4 are added to the set $\mathcal{F}$.

In Step 5, we delete the paths through the vertex of the function of $F = \{f'_2, f'_4\}$ from $\mathcal{P}$, and we obtain $P = \emptyset$.

In Step 6, since the condition of Case 2 holds, f_2 and f_4 in the modified BN (4.5) are replaced with f'_2 and f'_4 as follows:

$$\begin{cases} f_1 = x_1 \vee \neg x_4, \\ f_2 = x_2 \vee (\neg x_4 \wedge \neg x_5), \\ f_3 = x_3 \vee x_4, \\ f_4 = 1, \\ f_5 = x_4 \vee x_5, \end{cases}$$

and return to Step 1.

In Step 1, we substitute the constant function $x_4 = f_4 = 1$. As a result, we can obtain

$$\begin{cases} f_1 = x_1, \\ f_2 = x_2. \end{cases} \quad (4.6)$$

The interaction graph is shown in Fig. 4.2. In Step 2, any path cannot be generated (i.e., $\mathcal{P} = \emptyset$). Then, in Step 6, the condition of Case 3 holds, and the procedure is terminated. Therefore, we can obtain (4.6) as a reduced BN.

Finally, the original BN has four fixed points ($[0, 0, 1, 1, 1]$, $[0, 1, 1, 1, 1]$, $[1, 0, 1, 1, 1]$, $[1, 1, 1, 1, 1]$). The reduced BN has also four fixed points ($[0, 0]$, $[0, 1]$, $[1, 0]$, $[1, 1]$). Thus, fixed points are distinguishable by x_1 and x_2 . $\square$

From this example, we see that a given BN can be reduced while the interaction graph is changed.

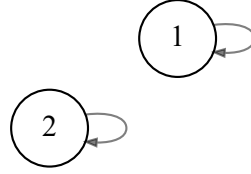


Figure 4.2: Interaction graph of the BN (4.6).

4.3 Numerical Example

In this section, we present a numerical example to validate the proposed method.

Consider 100 BNs with 20 states. Interaction graphs are randomly generated such that the scale-free property is satisfied for the in-degrees of the vertices. If there is a vertex that has no self-loop, then a self-loop is added. Logical operations are randomly assigned. In implementation, we use BDDs (binary decision diagrams). When the FVS-based model reduction method is applied to these BNs, the dimension of the state cannot be reduced. This is because the vertex sets of the interaction graphs for these BNs are an FVS.

We present the computation result using the proposed procedure. In 100 reduced BNs, the dimension of the states is derived as follows:

- The mean dimension: 18.35,
- The maximum dimension: 20,
- The minimum dimension: 7.

From these values, we see that the dimension of the state can be reduced using the proposed procedure. In addition, we present the information about $|\mathcal{N}_i|$ (the in-degree, that is, the number of elements of the state used in the Boolean function f_i). In 100 BNs, the max and mean of the maximum in-degree are as follows:

- Max: 18 (original), 11 (reduced),
- Mean: 3.13 (original), 1.99 (reduced).

From these values, we see that Boolean functions are simplified using the proposed procedure. In some methods of analysis and control of BNs, the computational complexity exponentially grows with the dimension of the state in each Boolean function (see, e.g. [40]). Hence, simplification of Boolean functions is important.

Finally, we comment on the computation time of the proposed procedure. For 100 BNs, the computation time of the proposed procedure is derived as follows:

- The mean computation time: 205 sec,

- The worst computation time: 11445 sec,

where we use Python 3.8.5 on the computer (CPU: Intel Core i7-4770K 3.50GHz, Memory: 8GB, OS: Windows 10). In 75 BNs of 100 BNs, the proposed method can be solved within 10 sec. In only a part of BNs, the long computation time is required. To derive the reduced model faster, it is future work to improve the proposed procedure.

4.4 Conclusion

In this chapter, we proposed a new method for model reduction of BNs. Focusing on the complement and absorption laws of Boolean functions, the dimension of the state is reduced based on the structure of the interaction graph of a given BN. The proposed method was demonstrated by a numerical example.

In future work, it is important to discuss the minimality of the reduced BN. In general, the dimension of the state in the reduced BN obtained by the proposed method is not minimal. Hence, it is important to find the structure of the interaction graph such that the reduced BN obtained by the proposed method is minimal. In the proposed method, the FVS-based model reduction method is utilized. It is also future work to develop a simpler procedure that the FVS-based model reduction method is not utilized.

Chapter 5

BDD-Based Control of Boolean Networks

5.1 Introduction

In this chapter, using BDDs, we propose a solution method for the finite-time control problem of BNs. The finite-time control problem is to find a control input sequence such that the state at a given final time is equal to the target value under a given initial state. This problem is one of the typical control problems in BNs [56]. Using the STP method, this problem has been studied as also the controllability problem. However, the STP method has difficulties for the same reasons explained in Section 2.5.2. Furthermore, in [56], it has been proven that the finite-time control problem for a general BN is NP-hard. In such cases, it is important to utilize efficient computation tools such as BDDs and SAT/SMT solvers. Since Boolean functions are simplified by using BDDs, it is appropriate to use BDDs for this problem. Although the data size of a BDD grows exponentially in the worst case, it provides a practically efficient solution in many cases.

In this chapter, using BDDs, we consider finding all combinations of the initial state and the control input sequence such that a certain control specification is satisfied (i.e., the initial state is not given in advance). Hence, the problem studied in this chapter is a more general problem including the problem in [56] as a special case.

This chapter is organized as follows. In Section 5.2, the problem studied here is formulated. In Section 5.3, a solution method using BDDs is proposed. A simple example is also presented to demonstrate the proposed method. In Section 5.4, the proposed method is applied to a 67-node BN. The STP method cannot be applied to such BNs. We compare the computation time using BDDs with that using the Z3 Theorem Prover. In Section 5.5, a problem is considered for extracting a solution that satisfies certain control specifications from the set of solutions obtained by the method in Section 5.3. In Section 5.6, we conclude this chapter, and address future efforts.

5.2 Problem Formulation

Consider the following finite-time control problem for the BN (2.3).

Problem 2. Suppose that for the BN (2.3), the target state $x^* \in \{0, 1\}^n$ and the final time T are given.

- i) Find all initial states such that $x(T) = x^* = [x_1^*, x_2^*, \dots, x_n^*]$ holds. Let $\mathcal{X}_0 \subseteq \{0, 1\}^n$ denote the obtained initial state set.
- ii) For each $x_0 \in \mathcal{X}_0$, find a control input sequence $U = [u(0), u(1), \dots, u(T-1)]$.

In the case where the initial state is given, this problem is one of the typical control problems for BNs, and has been studied in [56].

5.3 Proposed Solution Method

In this section, we propose a solution method for Problem 2. First, details of the proposed solution method is explained. Next, the proposed solution method is demonstrated by a simple example.

5.3.1 Solution Method Using BDDs

Consider solving Problem 2 using BDDs.

First, defining $f := [f_1, f_2, \dots, f_n]$, the final state $x(T)$ can be represented by

$$\begin{aligned} x(T) &= f(f(\dots f(f(x(0), u(0)), u(1)), \dots), u(T-1)) \\ &=: f^{(T)}(x(0), U), \end{aligned}$$

where the Boolean function $f^{(T)} : \{0, 1\}^n \times \{0, 1\}^{Tm}$ can be calculated from f . In addition, $f^{(T)}$ is represented as $f^{(T)} = [f_1^{(T)}, f_2^{(T)}, \dots, f_n^{(T)}]$. Then, Problem 2 can be rewritten as the following problem.

Problem 3. Find all combinations of the pair of the initial state $x(0)$ and the control input sequence U such that the following logical formula $F(x(0), U)$ is equal to 1 (true):

$$\begin{aligned} F(x(0), U) &:= (f_1^{(T)}(x(0), U) \leftrightarrow x_1^*) \wedge (f_2^{(T)}(x(0), U) \leftrightarrow x_2^*) \\ &\quad \wedge \dots \wedge (f_n^{(T)}(x(0), U) \leftrightarrow x_n^*), \end{aligned} \tag{5.1}$$

where “ $\leftrightarrow$ ” represents the logical equivalence operator.

To solve this problem, $F(x(0), U)$ is represented by a single BDD. Then, a solution to this problem can be easily derived from the obtained BDD.

Finally, we summarize the proposed procedure for solving Problem 3, which can be implemented on BDDs.

Procedure of solving Problem 3:

Step 1: Derive a BDD representing the Boolean function $f_i^{(T)}(x(0), U)$, $i = 1, 2, \dots, n$ by recursively substituting Boolean functions into other one. **Step 2:** Derive a BDD representing $F(x(0), U)$ of (5.1). **Step 3:** Find all combinations of $x(0)$ and U such that $F(x(0), U) = 1$.

Since Problem 3 is called an AllSAT (all solutions satisfiability) problem, it may be solved using a SAT/SMT (Satisfiability Modulo Theories) solver. However, using BDD,

we can obtain a compact representation, which will be useful in analysis and control of BNs¹.

5.3.2 Example

We present a simple example to explain the proposed procedure for solving Problem 3. Although the proposed procedure can be implemented on BDDs, we explain here the procedure using logical formulas.

Consider the following BN with three nodes and a single control input:

$$\begin{cases} x_1(t+1) = (x_1(t) \wedge x_3(t) \wedge u_1(t)) \vee x_2(t), \\ x_2(t+1) = x_3(t) \vee u_1(t), \\ x_3(t+1) = \neg x_1(t). \end{cases} \quad (5.2)$$

Suppose that the target state x^* and the final time T are given by $x^* = [1, 1, 0]$ and $T = 2$, respectively. Consider finding all combinations of the initial states and the control sequences such that $x(2) = [1, 1, 0]$ holds.

In Step 1, the Boolean function $f_i^{(2)}$ can be derived as follows. From $x_1(2) = (x_1(1) \wedge x_3(1) \wedge u_1(1)) \vee x_2(1)$ and (5.2), we can obtain

$$f_1^{(2)} = (\neg x_1(0) \wedge x_2(0) \wedge u_1(1)) \vee x_3(0) \vee u_1(0).$$

In a similar way, from $x_2(2) = x_3(1) \vee u_1(1)$, $x_3(2) = \neg x_1(1)$, and (5.2), the Boolean functions $f_2^{(2)}$ and $f_3^{(2)}$ can be obtained as

$$\begin{aligned} f_2^{(2)} &= \neg x_1(0) \vee u_1(1), \\ f_3^{(2)} &= \neg(x_1(0) \wedge x_3(0) \wedge u_1(0)) \wedge \neg x_2(0), \end{aligned}$$

respectively.

In Step 2, the Boolean function $F(x(0), U)$ is obtained as follows:

$$\begin{aligned} F(x(0), U) &= (f_1^{(2)} \leftrightarrow x_1^*) \wedge (f_2^{(2)} \leftrightarrow x_2^*) \wedge (f_3^{(2)} \leftrightarrow x_3^*) \\ &= \{(\neg x_1(0) \wedge x_2(0) \wedge u_1(1)) \vee x_3(0) \vee u_1(0)\} \wedge (\neg x_1(0) \vee u_1(1)) \\ &\quad \wedge \neg\{\neg(x_1(0) \wedge x_3(0) \wedge u_1(0)) \wedge \neg x_2(0)\} \\ &= (x_1(0) \wedge x_3(0) \wedge u_1(0) \wedge u_1(1)) \vee (\neg x_1(0) \wedge x_2(0) \wedge x_3(0)) \\ &\quad \vee (\neg x_1(0) \wedge x_2(0) \wedge u_1(0)) \vee (\neg x_1(0) \wedge x_2(0) \wedge u_1(1)) \\ &\quad \vee (x_2(0) \wedge x_3(0) \wedge u_1(1)) \vee (x_2(0) \wedge u_1(0) \wedge u_1(1)). \end{aligned}$$

The BDD representing $F(x(0), U)$ is shown in Fig. 5.1.

In Step 3, all combinations of the initial state and the control sequence that satisfy $F(x(0), U) = 1$ are found from Fig. 5.1. In this graph, the combinations of nodes that reach the terminal node 1 satisfies $F(x(0), U) = 1$. From observation of this graph, the combinations are obtained as

$$\begin{aligned} [x_1(0), x_2(0), x_3(0), u_1(0), u_1(1)] &= [0, 1, 0, 0, 1], [0, 1, 0, 1, *], [0, 1, 1, *, *], \\ &\quad [1, 0, 1, 1, 1], [1, 1, 0, 1, 1], [1, 1, 1, *, 1], \end{aligned}$$

¹A BDD-based SMT solver has been developed (see, e.g., [72]). In this chapter, we consider using only BDD techniques.

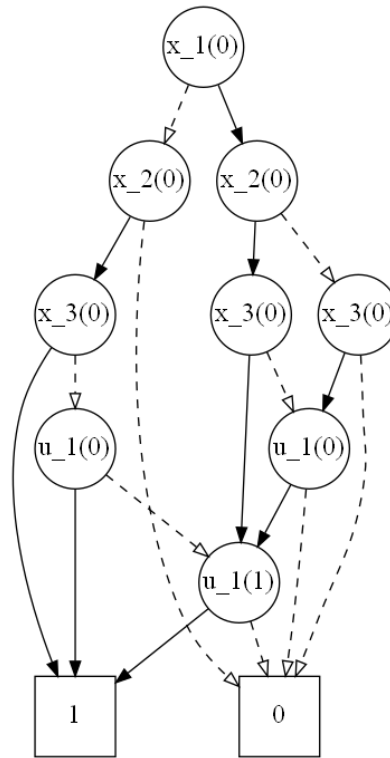


Figure 5.1: BDD of $F(x(0), U)$ in the example in Section 5.3.2. The nodes $x_1(0)$, $x_2(0)$, and $x_3(0)$ correspond to $x_1(0)$, $x_2(0)$, and $x_3(0)$, respectively. The nodes $u_1(0)$ and $u_1(1)$ correspond to $u_1(0)$ and $u_1(1)$, respectively.

where “*” means that the value can be given by any binary value. For example, $[0, 1, 1, *, *]$ means that when $x_0 = [0, 1, 1]$, the control input sequence can be taken any binary value (i.e., $[0, 0], [0, 1], [1, 0], [1, 1]$). Therefore, we see that there are 11 combinations. Thus, we can derive the solution of Problem 2.

5.4 Numerical Example

In this section, to demonstrate the proposed method, we present a numerical example of a BN such that the STP method cannot be applied.

Consider a 67-node, 5-input BN model of an apoptotic network [70]. The interactions between nodes are shown in Fig. 5.2, where input nodes correspond to elements of the control input. Suppose that the target state $x^* \in \{0, 1\}^{67}$ is given as the following binary vector:

$$x^* = [0_{16}, 1, 0_{16}, 1, 0_3, 1_{10}, 0, 1, 0_7, 1_3, 0_7, 1].$$

We consider the cases of $T = 1, 2, \dots, 15$. We remark here that the STP method, which is frequently used in analysis and control of BNs, cannot be applied to such BNs. This is because in the STP method, matrices with $2^{67} \times 2^{67+5}$ must be generally handled, and

Table 5.1: The computation result and the computation time for each T .

T	The number of combinations of $x(0)$ and U that reach x^*	Computation time of the proposed method (sec.)	Computation time of the SMT solver (sec.)
1	49152	0.00	0.09
2	27525120	0.00	0.15
3	812646400	0.02	1.39
4	26817331200	0.04	4.41
5	236165529600	0.20	31.08
6	920466227200	0.32	74.45
7	2460580577280	1.43	315.85
8	11803998289920	3.74	503.08
9	60487995752448	10.86	5792.54
10	229649820942336	23.16	10000 <
11	1089591461281792	49.62	10000 <
12	4360614260506624	83.34	10000 <
13	16290233280626688	120.56	10000 <
14	48994655819268096	167.57	10000 <
15	147099776955973632	227.29	10000 <

handling such matrices is hard on a conventional computer. See, e.g., [22] and [40] for details of the STP method. In this numerical example, we compare the computation time of the proposed procedure with that of the SMT solver.

We present the computation result. Table 5.1 shows the number of combinations of $x(0)$ and U , the computation time of the proposed procedure, and that of the SMT solver, where we use Python 3.9.7 on the computer (CPU: AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz, Memory: 32.0 GB, OS: Windows 11). We also use z3-solver 4.12.2.0 as the SMT solver. We set the timeout to 10000 sec. From Table 5.1, we see that the computation time of the proposed procedure is faster than that of the SMT solver.

We present one sample of combinations in the case of $T = 10$. As one of combinations, we can obtain the following initial state and control input sequence:

$$\begin{aligned}
x(0) &= [0_5, 1, 0_{10}, 1, 0_3, 1, 0_{12}, 1, 0_3, 1_3, 0, 1, 0_2, 1_2, \\
&\quad 0_3, 1_3, 0_3, 1_5, 0_2, 1, 0_3, 1], \\
u(0) &= [1, 0, 0, 0, 1], \\
u(1) &= u(2) = [1, 0, 0, 0, 0], \\
u(3) &= [0, 0, 0, 0, 0], \\
u(4) &= [0, 0, 0, 0, 1], \\
u(5) &= [0, 0, 0, 0, 0], \\
u(6) &= [1, 0, 0, 0, 0], \\
u(7) &= [0, 0, 0, 0, 0], \\
u(8) &= [1, 0, 0, 0, 0], \\
u(9) &= [0, 0, 0, 0, 0],
\end{aligned} \tag{5.3}$$

Instead of time response of the state, we present the Hamming distance defined by $\sum_{i=1}^{67} |x_i(k) - x_i^*(k)|$. Fig. 5.3 shows the Hamming distance at each time. From this figure, we see that $x(T) = x^*$ holds.

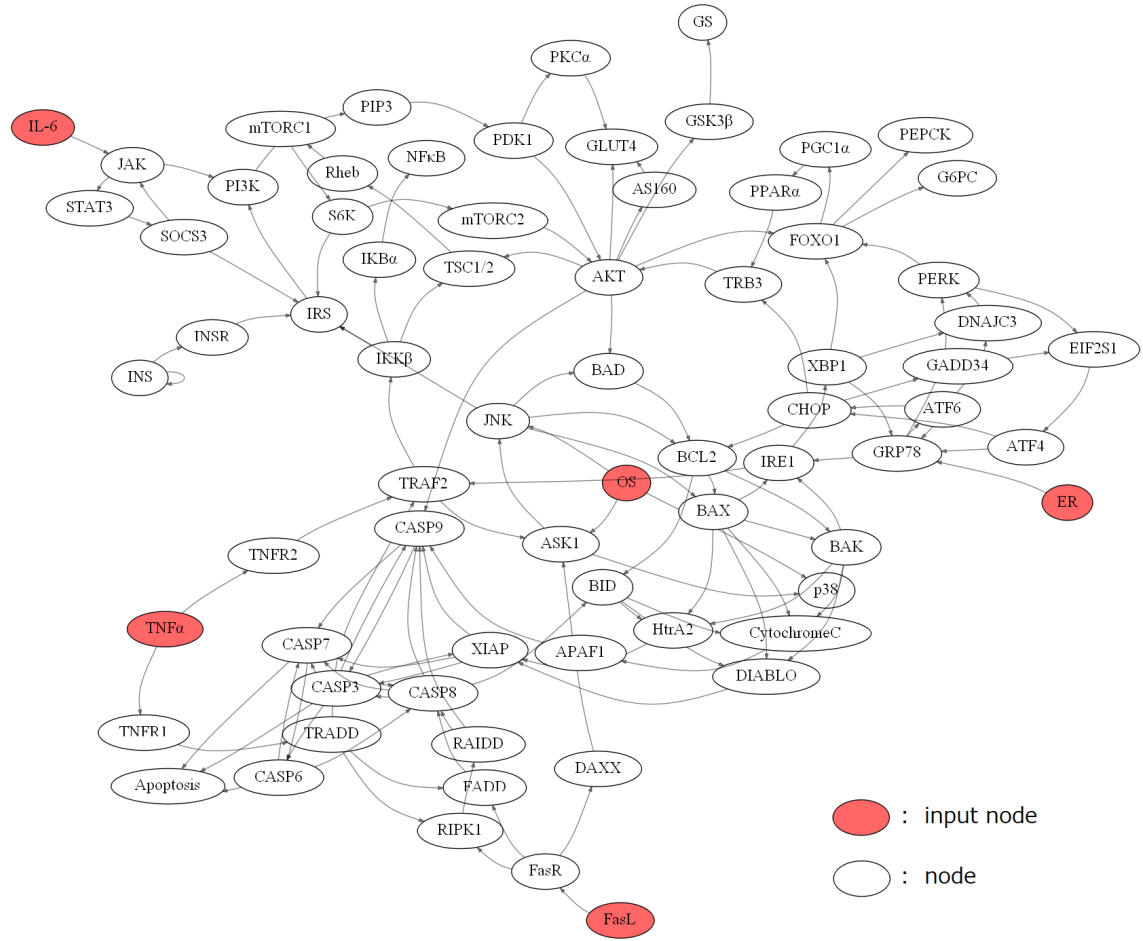


Figure 5.2: Interaction graph in the example in Section 5.4, where input nodes correspond to elements of the control input.

Finally, we further discuss the case where the initial state is given by (5.3). In this case, there are other control input sequences such that $x(T) = x^*$ holds. All control input sequences such that $x(T) = x^*$ holds can be characterized as follows:

$$\begin{aligned}
 u(0) &= [*, 0, 0, 0, 1], \\
 u(1) &= [*, 0, 0, 0, *], \\
 u(2) &= [0, 0, 0, 0, 0], \\
 u(3) &= [*, 0, 0, 0, *], \\
 u(4) &= [*, 0, 0, 0, 1], \\
 u(5) &= u(6) = [*, 0, 0, 0, 0], \\
 u(7) &= [*, 0, 0, 0, *], \\
 u(8) &= u(9) = [*, 0, 0, 0, 0].
 \end{aligned}$$

Hence, there are many patterns of the control input sequence such that $x(T) = x^*$ holds. For example, when it is desirable that the control input is zero as well as possible, we may

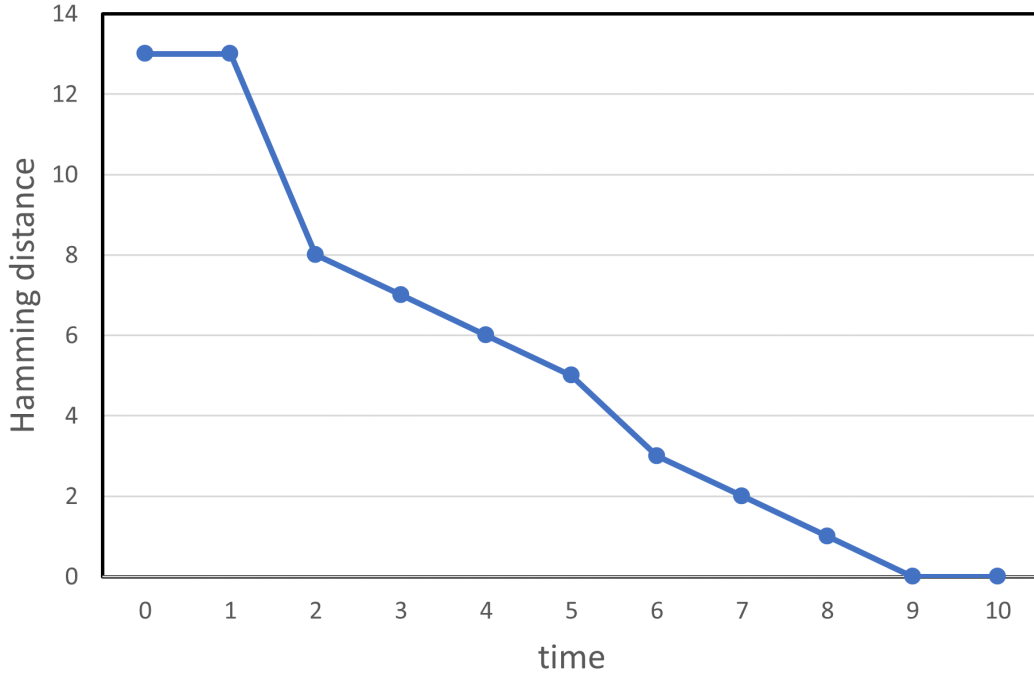


Figure 5.3: Time response of the Hamming distance.

set that $u_5(0)$ and $u_5(4)$ are equal to 1, and other inputs are zero. Thus, by the proposed method, we can obtain useful information.

5.5 Control problem that satisfies control specifications

Using the method presented in Section 5.3, a BDD that represents the solution of the finite-time control problem is obtained. The next step is to extract from this solutions those that satisfy certain specifications. This section provides an example.

5.5.1 Problem Formulation

In controlling gene regulatory networks, it is desirable to change the values of the control inputs as little as possible. Consider the following control problem.

Problem 4. A BN with n nodes and m control inputs, the target state $x^* \in \{0, 1\}^n$, and the final time T are given. For this BN, suppose that a BDD representing $F(x(0), U)$ (the solution to Problem 3) has already been obtained using the method in Section 5.3. The BDD of $F(x(0), U)$ denotes the initial state set that can reach the target state. One initial state x_0 included in the initial state set is given. Under the above conditions, among the control input sequences $U = [u(0), u(1), \dots, u(T-1)]$ that reach the target state x^* from the initial state x_0 in time T , find the one that maximizes the number of control inputs u_i , $i \in \{1, 2, \dots, m\}$ whose values do not change during time T .

5.5.2 Solution

First, we substitute the initial state $x(0) = x_0$ into $F(x(0), U)$ to obtain $F(U)$. This computation is performed on the BDD. This BDD represents the set of control input sequences that achieve stabilization under this initial state. We need to find a path on this BDD that reaches the terminal node 1 and that is a solution to Problem 4. A simple solution is to enumerate and search all possible paths, but since the total number of paths in a BDD is generally huge, a more efficient method is needed.

In this method, we attempt to efficiently obtain a solution to Problem 4 by expressing the path in a compressed form. Let $P = [p_1, p_2, \dots, p_m]$ denote the vector assigned to the nodes of the BDD of $F(U)$. P for a node represents multiple paths from the root node to that node. p_i , $i \in \{1, 2, \dots, m\}$ has four possible states, which are expressed as follows and have the following meanings:

- 0 : The value of u_i is 0 for all paths in P .
- 1 : The value of u_i is 1 for all paths in P .
- : The value of u_i is not fixed to either 0 or 1 for any path in P .
- * : The value of u_i can be either 0 or 1 for all paths in P .

Here is an example of how to read P . Suppose the number of control inputs is $m = 2$, the final time is $T = 2$, and the terminal node 1 of the BDD of $F(U)$ has

$$P = [1, *].$$

This P represents the following four paths:

$$\begin{aligned} [u_1(0), u_2(0), u_1(1), u_2(1)] = & [1, 0, 1, 0], [1, 0, 1, 1], \\ & [1, 1, 1, 0], [1, 1, 1, 1] \end{aligned} \quad (5.4)$$

where the value of u_1 is fixed at 1, and u_2 can take any value at each time. In this way, when P has states such as "-" or "*", P represents multiple paths.

When terminal node 1 has P , the maximum number of control inputs whose values do not change in the control sequence represented by P is as follows:

$$m - |\{p \in P \mid p = -\}| \quad (5.5)$$

For example, for P in the example (5.4), the maximum number of control inputs whose values do not change is calculated to be $2 - 0 = 2$. The paths are $[1, 0, 1, 0]$ where $u_1 = 1$, $u_2 = 0$, and $[1, 1, 1, 1]$ where $u_1 = 1$, $u_2 = 1$. From the above, we can see that a single P can represent multiple routes, and that the solution to Problem 4 can be easily found from the set of P held by terminal node 1. This allows for a more efficient solution than enumerating all the paths.

We explain the solution to Problem 4. In this method, the BDD of $F(U)$ is searched from the root node using breadth-first search, and all paths that can reach each node are found sequentially, and finally all paths that can reach the terminal node 1 are obtained. In breadth-first search, we start from the root node and search downwards, preferentially searching nodes of the same depth. In BDD, the order of variable names is fixed, and

nodes at the same depth have the same variable names. Therefore, in a breadth-first search of a BDD, we assign values to each variable in the order defined by the BDD.

Assign labels $e_1, e_2, \dots$ to each node of the BDD of $F(U)$. Let the label of the root node be e_1 , and the label of terminal node 1 be e_{t1} . Let $\mathcal{A}(e)$ be the set of P that node e has. Initially, all of them are empty. Let $o = [v_1, v_2, \dots] \subseteq \{u_i(t) \mid i = 1, \dots, m, t = 0, \dots, T-1\}$ denote the vector representing the order of variable name defined in the BDD. The proposed procedure is presented as follows.

Procedure of solving Problem 4:

Step 1: Place $P = [* , \dots , *]$ at the root node of BDD of $F(U)$ ($\mathcal{A}(e_1) := \{[* , * , \dots , *]\}$).

Step 2: Follow the following conditional branch.

- If $o \neq \emptyset$, Let the variable name at the beginning of o be called v , and delete v from o .
- If $o = \emptyset$, then go to Step 4.

Step 3: For each node e_k with variable name v , let the child nodes in the 0-direction and 1-direction be e_{k0} and e_{k1} , respectively. Let $\mathcal{A}^0(e_k)$ and $\mathcal{A}^1(e_k)$ be as copies of $\mathcal{A}(e_k)$. For all $P \in \mathcal{A}^0(e_k)$, follow the following conditional branching.

- If $p_{v(e_k)} = *$, then set $p_{v(e_k)} := 0$.
- If $p_{v(e_k)} = 1$, then set $p_{v(e_k)} := -$.
- If $p_{v(e_k)} = 0$ or $-$, do nothing.

For all $P \in \mathcal{A}^1(e_k)$, follow the following conditional branching.

- If $p_{v(e_k)} = *$, then set $p_{v(e_k)} := 1$.
- If $p_{v(e_k)} = 0$, then set $p_{v(e_k)} := -$.
- If $p_{v(e_k)} = 1$ or $-$, do nothing.

Update the path set $\mathcal{A}(e_{k0}) := \mathcal{A}(e_{k0}) \cup \mathcal{A}^0(e_k)$, $\mathcal{A}(e_{k1}) := \mathcal{A}(e_{k1}) \cup \mathcal{A}^1(e_k)$. Go to Step 3.

Step 4: For $P \in \mathcal{A}(e_{t1})$, choose the state with the smallest number of "-". This is the solution to Problem 4.

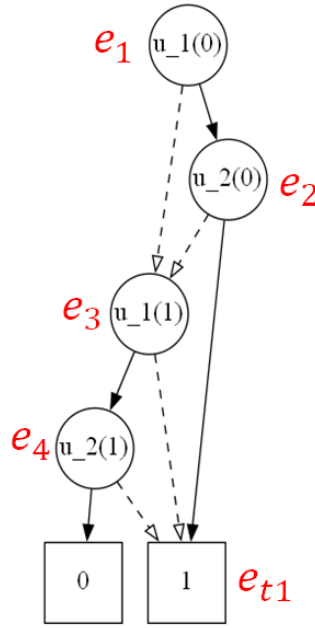
We explain each step.

In Step 1, the vector P is assigned to the root node. At this point, P represents all possible paths, and the candidates are narrowed down in Steps 2 and 3.

In Step 2, we determine the variable names of the nodes to be searched in Step 3. This is equivalent to performing a breadth-first search on the BDD. If all variable names have been searched, go to Step 4.

In Step 3, the nodes having the variable name determined in Step 2 pass its own P to its child node. The branch selection at this time is reflected in P . Then return to Step 2. By repeating steps 2 and 3, P is propagated from the root node to terminal node 1.

In Step 4, the solution to Problem 4 is found from P of the terminal node 1.

Figure 5.4: BDD of $F(U)$ in the example in Section 5.5.3.

5.5.3 Example

We present a simple example to explain this procedure.

Let $m = 2$, $T = 2$. Suppose that $F(U)$ is obtained by substituting some initial state for the given $F(x(0), U)$. The BDD of $F(U)$ is shown in Fig. 5.4. The ordering of the variable names in this BDD is $o = [u_1(0), u_2(0), u_1(1), u_2(1)]$.

First, in Step 1, we assign $P = \{[*], [*]\}$ to the root node e_1 .

$$\mathcal{A}(e_1) := \{[*], [*]\}$$

In Step 2, extract the variable name from the beginning of o and call it v .

$$v := u_1(0), \quad o := [u_2(0), u_1(1), u_2(1)]$$

In Step 3, search for child nodes from node e_1 with variable name $v = u_1(0)$. Let $\mathcal{A}^0(e_1)$ and $\mathcal{A}^1(e_1)$ be as copies of $\mathcal{A}(e_1)$. $\mathcal{A}^0(e_1)$ reflects $u_1(0) = 0$, and $\mathcal{A}^1(e_1)$ reflects $u_1(0) = 1$.

$$\mathcal{A}^0(e_1) := \{[0, *]\}, \quad \mathcal{A}^1(e_1) := \{[1, *]\}$$

Then, $\mathcal{A}^0(e_1)$ and $\mathcal{A}^1(e_1)$ are assigned to the child nodes e_3 , e_2 of e_1 , respectively. Then, go to Step 2.

$$\begin{aligned} \mathcal{A}(e_3) &:= \mathcal{A}(e_3) \cup \mathcal{A}^0(e_1) & \mathcal{A}(e_2) &:= \mathcal{A}(e_2) \cup \mathcal{A}^1(e_1) \\ &= \{[0, *]\}, & &= \{[1, *]\} \end{aligned}$$

In Step 2, extract the variable name from the beginning of o and call it v .

$$v := u_2(0), \quad o := [u_1(1), u_2(1)]$$

In Step 3, search for child nodes from node e_2 with variable name $v = u_2(0)$. Let $\mathcal{A}^0(e_2)$ and $\mathcal{A}^1(e_2)$ be as copies of $\mathcal{A}(e_2)$. $\mathcal{A}^0(e_2)$ reflects $u_2(0) = 0$, and $\mathcal{A}^1(e_2)$ reflects $u_2(0) = 1$.

$$\mathcal{A}^0(e_2) := \{[1, 0]\}, \quad \mathcal{A}^1(e_2) := \{[1, 1]\}$$

Then, $\mathcal{A}^0(e_2)$ and $\mathcal{A}^1(e_2)$ are assigned to the child nodes e_3 , e_{t1} of e_2 , respectively. Then, go to Step 2.

$$\begin{aligned} \mathcal{A}(e_3) &:= \mathcal{A}(e_3) \cup \mathcal{A}^0(e_2) & \mathcal{A}(e_{t1}) &:= \mathcal{A}(e_{t1}) \cup \mathcal{A}^1(e_2) \\ &= \{[0, *], [1, 0]\}, & &= \{[1, 1]\} \end{aligned}$$

In Step 2, extract the variable name from the beginning of o and call it v .

$$v := u_1(1), \quad o := [u_2(1)]$$

In Step 3, search for child nodes from node e_3 with variable name $v = u_1(1)$. Let $\mathcal{A}^0(e_3)$ and $\mathcal{A}^1(e_3)$ be as copies of $\mathcal{A}(e_3)$. $\mathcal{A}^0(e_3)$ reflects $u_1(1) = 0$, and $\mathcal{A}^1(e_3)$ reflects $u_1(1) = 1$.

$$\mathcal{A}^0(e_3) := \{[0, *], [-, 0]\}, \quad \mathcal{A}^1(e_3) := \{[-, *], [1, 0]\}$$

Then, $\mathcal{A}^0(e_3)$ and $\mathcal{A}^1(e_3)$ are assigned to the child nodes e_{t1} , e_4 of e_3 , respectively. Then, go to Step 2.

$$\begin{aligned} \mathcal{A}(e_{t1}) &:= \mathcal{A}(e_{t1}) \cup \mathcal{A}^0(e_3) & \mathcal{A}(e_4) &:= \mathcal{A}(e_4) \cup \mathcal{A}^1(e_3) \\ &= \{[1, 1], [0, *], [-, 0]\}, & &= \{[-, *], [1, 0]\} \end{aligned}$$

In Step 2, extract the variable name from the beginning of o and call it v .

$$v := u_2(1), \quad o := \emptyset$$

In Step 3, search for child nodes from node e_4 with variable name $v = u_2(1)$. Let $\mathcal{A}^0(e_4)$ and $\mathcal{A}^1(e_4)$ be as copies of $\mathcal{A}(e_4)$. $\mathcal{A}^0(e_4)$ reflects $u_2(1) = 0$, and $\mathcal{A}^1(e_4)$ reflects $u_2(1) = 1$.

$$\mathcal{A}^0(e_4) := \{[-, 0], [1, 0]\}, \quad \mathcal{A}^1(e_4) := \{[-, 1], [1, -]\}$$

Then, $\mathcal{A}^0(e_4)$ is assigned to the child nodes e_{t1} of e_3 (there is no need to consider assigning it to the terminal node 0). Then, go to Step 2.

$$\begin{aligned} \mathcal{A}(e_{t1}) &:= \mathcal{A}(e_{t1}) \cup \mathcal{A}^0(e_4) \\ &= \{[1, 1], [0, *], [-, 0], [1, 0]\} \end{aligned}$$

In Step 2, since $o = \emptyset$, go to Step 4.

In Step 4, we search for a path from $\mathcal{A}(e_{t1})$ that is a solution to Problem 4. Note that there are five paths to the terminal node 1 on the BDD, and they can be represented by four P . P that maximize equation (5.4) are

$$[1, 1], [0, *], [1, 0].$$

From this, we obtain the following solution to Problem 4:

$$\begin{aligned} [u_1(0), u_2(0), u_1(1), u_2(1)] &= [0, 0, 0, 0], [0, 1, 0, 1], \\ &\quad [1, 0, 1, 0], [1, 1, 1, 1]. \end{aligned}$$

5.6 Conclusion

In this chapter, based on BDDs, we proposed a solution method for the finite-time control problem of BNs. Using the proposed method, we can obtain all combinations of the initial state and the control input sequence such that the state at the final time reaches the target state. The effectiveness of the proposed method was presented by a BN with 67-node and 5-input.

In future work, using the BDD obtained by the proposed method, we will consider developing a method for deriving both the initial state and the control input sequence such that another control specification is satisfied. Moreover, as an extension of the proposed method, it is also significant to develop a solution method for the optimal control problem using BDDs.

Chapter 6

Conclusions

This thesis has presented reduction methods and a control method of BN using BDD. Since BNs generally have a huge number of states, efficient methods are required for their control and analysis. In these studies, states are expressed in a compressed form, BDDs, which allows some traditional methods to be applied to large BNs. Summarize the findings from each chapter.

In Chapter 3, a model reduction method of BNs using BDDs has been proposed. It is based on the existing method [45, 47], but this method cannot be simply replicated using BDDs. By using the interaction graph expressing interactions between elements of the state, all other required computations can be performed on the BDD. By this method, a given BN is reduced to a BN with the number of elements of the FVS. The effectiveness of the method is presented by numerical examples on large-scale BNs that the STP method cannot be applied.

In Chapter 4, a reduction method to further reduce the reduced model proposed in Chapter 3 has been proposed. As a reduction approach different from existing methods [45, 47], we focus on the complement and absorption laws of Boolean functions. It is shown that patterns of relationships between the elements that lead to the computation of these laws can be found in interaction graphs. Although this method is inefficient in terms of reduction performance and the amount of calculation, it is sufficient as a method to apply to the reduced model obtained in Chapter 3. In the calculation examples, it is showed that this method is effective for BNs that cannot be further reduced by the method in Chapter 3. However, this reduced model is also not a minimal model, so it is a future study to consider how to obtain a minimal model.

In Chapter 5, a finite-time control method for BNs using BDDs has been proposed. The finite-time control problem is to find all sequences of initial states and control input sequences that can reach the target state at a given final time. Although the number of solutions becomes enormous, we have succeeded in efficiently representing them using BDDs. In the calculation examples, the effectiveness of this method was demonstrated by comparison with an SMT solver. In addition, we consider the problem of extracting a solution that satisfies a control specification from the set of solutions obtained by this method. It is shown that it is possible to efficiently find a solution from solutions expressed in BDD while taking advantage of its compression capability. Future work includes considering the application of various other control specifications.

Acknowledgements

I sincerely thank my advisor Professor Koich Kobayashi for the proper guidance of and numerous suggestions for this work.

I would like to thank Professor Yuh Yamashita, Professor Satoshi Kanai for being my committee.

I would like to thank Professor Yuh Yamashita for being my committee and his good advise.

Special thanks go to students of Dynamical Systems and Control Laboratory for my pleasant days in the laboratory.

I also would like to express my gratitude all the staff and students whom I have met during my doctoral life at Hokkaido university.

Finally, I would like to thank my family for their support that have enabled me to complete my Ph.D. degree.

References

- [1] Ting Chen, Hongyu L He, and George M Church. Modeling gene expression with differential equations. In *Biocomputing'99*, pages 29–40. World Scientific, 1999.
- [2] Thomas Mestl, Erik Plahte, and Stig W Omholt. A mathematical framework for describing and analysing gene regulatory networks. *Journal of theoretical Biology*, 176(2):291–300, 1995.
- [3] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons, 2005.
- [4] René Thomas. Boolean formalization of genetic control circuits. *Journal of theoretical biology*, 42(3):563–585, 1973.
- [5] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In *Biocomputing'99*, pages 17–28. World Scientific, 1999.
- [6] Min Zou and Suzanne D Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.
- [7] Hiroshi Matsuno, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Hybrid Petri net representation of gene regulatory network. In *Biocomputing 2000*, pages 341–352. World Scientific, 1999.
- [8] L Jason Steggles, Richard Banks, Oliver Shaw, and Anil Wipat. Qualitatively modelling and analysing genetic regulatory networks: a petri net approach. *Bioinformatics*, 23(3):336–343, 2007.
- [9] Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theoretical Biol.*, 22(3):437–467, 1969.
- [10] B Faryabi, A Datta, and ER Dougherty. On approximate stochastic control in genetic regulatory networks. *IET Systems Biology*, 1(6):361–368, 2007.
- [11] Ettore Fornasini and Maria Elena Valcher. Fault detection of Boolean control networks. In *53rd IEEE Conference on Decision and Control*, pages 6542–6547, 2014.
- [12] Zhihua Zhang, Thomas Leifeld, and Ping Zhang. Active fault detection of Boolean control networks. In *2018 Annual American Control Conference (ACC)*, pages 5001–5006. IEEE, 2018.

- [13] David G Green, Tania G Leishman, and Suzanne Sadedin. The emergence of social consensus in Boolean networks. In *2007 IEEE Symposium on Artificial Life*, pages 402–408, 2007.
- [14] Yuhu Wu and Tielong Shen. A logical dynamical systems approach to modeling and control of residual gas fraction in ic engines. In *7th IFAC Symposium on Advances in Automotive Control*, pages 495–500, 2013.
- [15] Koichi Kobayashi, Jun-Ichi Imura, and Kunihiro Hiraishi. Polynomial-time algorithm for controllability test of a class of Boolean biological networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2010(1):210685, 2010.
- [16] Shun-ichi Azuma, Takahiro Yoshida, and Toshiharu Sugie. Structural monostability of activation-inhibition Boolean networks. *IEEE Transactions on Control of Network Systems*, 4(2):179–190, 2017.
- [17] Shuqi Chen, Yuhu Wu, Matthew Macauley, and Xi-Ming Sun. Monostability and bistability of Boolean networks using semitensor products. *IEEE Transactions on Control of Network Systems*, 6(4):1379–1390, 2018.
- [18] Min Meng, James Lam, Jun-E Feng, and Kie Chung Cheung. Stability and stabilization of Boolean networks with stochastic delays. *IEEE Transactions on Automatic Control*, 64(2):790–796, 2018.
- [19] Yuqian Guo, Pan Wang, Weihua Gui, and Chunhua Yang. Set stability and set stabilization of Boolean control networks based on invariant subsets. *Automatica*, 61:106–112, 2015.
- [20] Rongjian Liu, Jianquan Lu, Yang Liu, Jinde Cao, and Zheng-Guang Wu. Delayed feedback control for stabilization of Boolean control networks with state delay. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3283–3288, 2017.
- [21] Amol Yerudkar, Carmen Del Vecchio, and Luigi Glielmo. Feedback stabilization control design for switched Boolean control networks. *Automatica*, 116:108934, 2020.
- [22] Daizhan Cheng and Hongsheng Qi. Controllability and observability of Boolean control networks. *Automatica*, 45(7):1659–1667, 2009.
- [23] Dmitriy Laschov and Michael Margaliot. Controllability of Boolean control networks via the perron–frobenius theory. *Automatica*, 48(6):1218–1223, 2012.
- [24] Qunxi Zhu, Yang Liu, Jianquan Lu, and Jinde Cao. Further results on the controllability of Boolean control networks. *IEEE Transactions on Automatic Control*, 64(1):440–442, 2018.
- [25] Ettore Fornasini and Maria Elena Valcher. Observability, reconstructibility and state observers of Boolean control networks. *IEEE Transactions on Automatic Control*, 58(6):1390–1401, 2013.
- [26] Yongyuan Yu, Min Meng, Jun-e Feng, and Ge Chen. Observability criteria for Boolean networks. *IEEE Transactions on Automatic Control*, 67(11):6248–6254, 2022.

REFERENCES

- [27] Koichi Kobayashi and Kunihiro Hiraishi. Optimal control of gene regulatory networks with effectiveness of multiple drugs: A Boolean network approach. *BioMed research international*, 2013, 2013.
- [28] Ettore Fornasini and Maria Elena Valcher. Optimal control of Boolean control networks. *IEEE Transactions on Automatic Control*, 59(5):1258–1270, 2014.
- [29] Yuhu Wu, Xi-Ming Sun, Xudong Zhao, and Tielong Shen. Optimal control of Boolean control networks with average cost: A policy iteration approach. *Automatica*, 100:378–387, 2019.
- [30] Qunxi Zhu, Yang Liu, Jianquan Lu, and Jinde Cao. On the optimal control of Boolean control networks. *SIAM Journal on Control and Optimization*, 56(2):1321–1341, 2018.
- [31] Ilya Shmulevich, Edward R Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.
- [32] Ilya Shmulevich, Ilya Gluhovsky, Ronaldo F Hashimoto, Edward R Dougherty, and Wei Zhang. Steady-state analysis of genetic regulatory networks modelled by probabilistic Boolean networks. *Comparative and functional genomics*, 4(6):601–608, 2003.
- [33] Ranadip Pal, Aniruddha Datta, Michael L Bittner, and Edward R Dougherty. Intervention in context-sensitive probabilistic Boolean networks. *Bioinformatics*, 21(7):1211–1218, 2005.
- [34] Fangfei Li and Jitao Sun. Controllability of probabilistic Boolean control networks. *Automatica*, 47(12):2765–2771, 2011.
- [35] Koichi Kobayashi and Kunihiro Hiraishi. An integer programming approach to optimal control problems in context-sensitive probabilistic Boolean networks. *Automatica*, 47(6):1260–1264, 2011.
- [36] Mitsuru Toyoda and Yuhu Wu. Mayer-type optimal control of probabilistic Boolean control network with uncertain selection probabilities. *IEEE Transactions on Cybernetics*, 51:3079–3092, 2021.
- [37] Antonio Acernese, Amol Yerudkar, Luigi Glielmo, and Carmen Del Vecchio. Reinforcement learning approach to feedback stabilization problem of probabilistic Boolean control networks. *IEEE Control Systems Letters*, 5(1):337–342, 2020.
- [38] Hiroaki Kitano. Computational systems biology. *Nature*, 420(6912):206, 2002.
- [39] Hiroaki Kitano. Cancer as a robust system: implications for anticancer therapy. *Nature Reviews Cancer*, 4(3):227–235, 2004.
- [40] Daizhan Cheng, Hongsheng Qi, and Zhiqiang Li. *Analysis and control of Boolean networks: A semi-tensor product approach*. London, U.K.: Springer, 2011.
- [41] Takayuki Ishizaki, Kenji Kashima, Jun-ichi Imura, and Kazuyuki Aihara. Model reduction and clusterization of large-scale bidirectional networks. *IEEE Transactions on Automatic Control*, 59(1):48–63, 2013.

- [42] Yu Kawano, Bart Besselink, Jacquélien MA Scherpen, and Ming Cao. Data-driven model reduction of monotone systems by nonlinear dc gains. *IEEE Transactions on Automatic Control*, 65(5):2094–2106, 2019.
- [43] X. Cheng and J.M.A. Scherpen. Model reduction methods for complex network systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):425–453, 2021.
- [44] Min Meng, James Lam, Jun-e Feng, and Xiuxian Li. l_1 -gain analysis and model reduction problem for Boolean control networks. *Information Sciences*, 348:68–83, 2016.
- [45] Koichi Kobayashi. Design of fixed points in Boolean networks using feedback vertex sets and model reduction. *Complexity*, 2019, 2019.
- [46] Aurélien Naldi, Elisabeth Remy, Denis Thieffry, and Claudine Chaouiya. A reduction of logical regulatory graphs preserving essential dynamical properties. In *International Conference on Computational Methods in Systems Biology*, pages 266–280. Springer, 2009.
- [47] Alan Veliz-Cuba. Reduction of Boolean network models. *Journal of theoretical biology*, 289:167–172, 2011.
- [48] Yongyi Yan, Jumei Yue, and Zengqiang Chen. Algebraic method of simplifying Boolean networks using semi-tensor product of matrices. *Asian Journal of Control*, 21(6):2569–2577, 2019.
- [49] Randal E Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986.
- [50] Giang Van Trinh, Tatsuya Akutsu, and Kunihiro Hiraishi. An FVS-based approach to attractor detection in asynchronous random Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(2):806–818, 2022.
- [51] Qixia Yuan, Hongyang Qu, Jun Pang, and Andrzej Mizera. Improving BDD-based attractor detection for synchronous Boolean networks. *Science China Information Sciences*, 59(8):1–16, 2016.
- [52] Tatsuya Akutsu, Yang Zhao, Morihiro Hayashida, and Takeyuki Tamura. Integer programming-based approach to attractor detection and control of Boolean networks. *IEICE Transactions on Information and Systems*, 95(12):2960–2970, 2012.
- [53] Tomoya Mori and Tatsuya Akutsu. Attractor detection and enumeration algorithms for Boolean networks (Mini Review). *Computational and Structural Biotechnology Journal*, 20:2512–2520, 2022.
- [54] Takeyuki Tamura and Tatsuya Akutsu. Detecting a singleton attractor in a Boolean network utilizing SAT algorithms. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(2):493–501, 2009.
- [55] Koichi Kobayashi and Kunihiro Hiraishi. ILP/SMT-based method for design of Boolean networks based on singleton attractors. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(6):1253–1259, 2014.

REFERENCES

- [56] Tatsuya Akutsu, Morihiro Hayashida, Wai-Ki Ching, and Michael K Ng. Control of Boolean networks: Hardness results and algorithms for tree structured networks. *Journal of Theoretical Biology*, 244(4):670–679, 2007.
- [57] Madalena Chaves. Methods for qualitative analysis of genetic networks. In *2009 European Control Conference*, pages 671–676, 2009.
- [58] Laurent Tournier and Madalena Chaves. Uncovering operational interactions in genetic networks using asynchronous Boolean dynamics. *Journal of Theoretical Biology*, 260(2):196–209, 2009.
- [59] Elisabeth Remy, Paul Ruet, Luis Mendoza, Denis Thieffry, and Claudine Chaouiya. From logical regulatory graphs to standard petri nets: Dynamical roles and functionality of feedback circuits. In *Transactions on Computational Systems Biology VII*, pages 56–72. Springer, 2006.
- [60] Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on computers*, 45(9):993–1002, 1996.
- [61] Shin-ichi Minato, Nagisa Ishiura, and Shuzo Yajima. Shared binary decision diagram with attributed edges for efficient Boolean function manipulation. In *Proceedings of the 27th ACM/IEEE design automation conference*, pages 52–57, 1991.
- [62] Eric Felt, Gary York, Robert Brayton, and Alberto Sangiovanni-Vincentelli. Dynamic variable reordering for BDD minimization. In *Proceedings of EURO-DAC 93 and EURO-VHDL 93-European Design Automation Conference*, pages 130–135. IEEE, 1993.
- [63] Fadi A Aloul, Igor L Markov, and Karem A Sakallah. MINCE: A static global variable-ordering heuristic for SAT search and BDD manipulation. *J. Univers. Comput. Sci.*, 10(12):1562–1596, 2004.
- [64] Guy Even, J Seffi Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.
- [65] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
- [66] Yufei Xiao and Edward R Dougherty. The impact of function perturbations in Boolean networks. *Bioinformatics*, 23(10):1265–1273, 2007.
- [67] Wenpin Hou, Takeyuki Tamura, Wai-Ki Ching, and Tatsuya Akutsu. Finding and analyzing the minimum set of driver nodes in control of Boolean networks. *Advances in Complex Systems*, 19(03), 2016.
- [68] Wenpin Hou, Peiying Ruan, Wai-Ki Ching, and Tatsuya Akutsu. On the number of driver nodes for controlling a Boolean network when the targets are restricted to attractors. *Journal of theoretical biology*, 463:1–11, 2019.
- [69] Stuart A Kauffman et al. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

- [70] Pritha Dutta, Lichun Ma, Yusuf Ali, Peter Sloot, and Jie Zheng. Boolean network modeling of β -cell apoptosis and insulin resistance in type 2 diabetes mellitus. *BMC Systems Biology*, 13(2):1–12, 2019.
- [71] Xiaoqing Cheng, Takeyuki Tamura, Wai-Ki Ching, and Tatsuya Akutsu. Discrimination of singleton and periodic attractors in Boolean networks. *Automatica*, 84:205–213, 2017.
- [72] Martin Jonáš and Jan Strejček. Q3B: an efficient BDD-based SMT solver for quantified bit-vectors. In *International Conference on Computer Aided Verification*, pages 64–73. Springer, 2019.